Stewart Platform with Electronics Control and Leap Motion Interaction

Tim Branch
Henry Liu
Don Nguyen

Project Sponsor:
Progressive Automations

ENPH 459
Engineering Physics
The University of British Columbia

June 28, 2018

Table of Contents

List of Figures and Tables

1 Introduction

1.1 Project Objectives

The objective of this project is to demonstrate the capabilities of linear actuators manufactured by Progressive Automations. This, once the platform is functional, requires the design and development of one or more demos. These demos must provide a creative and interesting motive in controlling the platform and be included as a part of the end product.

The final implementation of the platform must be appropriate for presentation, and therefore needs to operate at a smooth, reasonable speed while being able to approximate the theoretical range of motion (enough to properly run the demos). For the maximal user experience, the platform should be responsive to external control via joystick or Leap Motion, which needs to be accurate to the motion of the platform and perform with near-zero latency. The control method should require minimal setup and explanation for the end user.

Finally, any external platform components including electronics, controllers, and the power supply must fit in a portable enclosure in order for the platform to be mobile enough for transportation to/from trade shows. The enclosure should have easy access to the Arduino board for testing and modification, have plug-and-play connection capabilities to peripherals such as the actuator cables, and ideally fit within the platform base when active.

1.2 Scope and Limitations

This report outlines the work completed during the period of October 2017 to April 2018 in implementing the following major tasks:

- improving serial communication and responsiveness, through reworking existing Arduino code and researching micro-controller alternatives;
- reworking and testing Leap Motion code with the platform;
- creating a user interface for interaction with the platform and Leap Motion controller;
- designing and fabricating new circuit boards and wiring for the electronics;
- designing and fabricating an enclosure for all electronic components; and
- designing and fabricating a tilt maze demo.

The project uses the linear actuators (PA-14P) and motor controllers (Multimoto) originally supplied by the sponsor. Though there was the option of looking at different product options provided by Progressive Automations, there was little motivation as the current actuator model provided the best speed at its size. Alternatives to the included potentiometer such as an accelerometer or a Hall effect sensor were not tested; it is unknown if implementing those would have reduced noise in the feedback system, at the cost of a more complex control method.

Joysticks and game controllers were briefly investigated as additional user input devices; however, due to time constraints and disparate control implementations they were ultimately not considered. The Leap Motion controller was also tested at an early stage of the project and found to be usable, which also relieved concerns that a back-up form of control would be necessary.

Since the user interface was designed in Qt, the software will be released under the LGPL (GNU Lesser General Public License) and must be kept as an open-source project while without a commercial Qt license. This should not be a concern given that the project is not meant for sale or as private IP.

1.3 Organization

The following sections of the report are outlined as below:

Discussion

- Encompasses an overview of the theory as well as the methods and testing required to implement the project deliverables.
- Explains in detail the reasoning and design process for each project component.
- Includes the evaluation and analysis of the final results.

Conclusion

- Provides a meaningful interpretation and inferences of the results from the discussion.
- Elaborates on how the results satisfy the project objectives.

Project Deliverables

- Lists the deliverables/items and their state when they are handed off to the sponsor at the time of project completion.
- Identifies the major financial costs of the projects and where funds have been allocated.
- States any ongoing commitments and contributions from team members after the project completion date.

Recommendations

- Outlines additional suggested actions based on the conclusion in order for the sponsor to continue work on the project.

Appendices

- Provides supplemental technical information and data as further detail of the content in the report, and also as a sample of the content/documentation found in the project repository[1].

[1] `https://github.com/henrymliu/StewartPlatform`

2 Discussion

2.1 Theory

2.1.1 Stewart Platform Kinematics

For a platform in space, there are six degrees of freedom: three for position, three for direction. To fully control all of these degrees of freedom, six independent degrees of control are needed (in this case, actuators or motors). To ensure the platform was rigid with no extraneous degrees of freedom throughout its range of motion, the previous team designed the Stewart platform to consist of two triangular-shaped plates, offset by sixty degrees from each other, with two actuators connected to each vertex, as shown in Figure 1.

Figure 1: Platform with actuators labeled.

To control the platform, one starts with the desired platform position – a six-vector containing the relative position and direction of the platform – and working backwards, it is possible to determine the vector in space that each actuator must be. This can be done by considering this simple formula [5]:

$$\mathbf{L_i} = \mathbf{R_{ZYX}} \mathbf{T_i} + \mathbf{P} - \mathbf{B_i}$$

where $\mathbf{T_i}$ and $\mathbf{B_i}$ represent the position of connection of the i'th actuator on the top and bottom plates, respectively, and $\mathbf{L_i}$ represents the direction vector for the i'th actuator. As well,

$$\mathbf{R_{ZYX}} = \mathbf{R_Z} \mathbf{R_Y} \mathbf{R_X}$$

where

$$\mathbf{R_X} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & -\sin(\theta_x) \\ 0 & \sin(\theta_x) & \cos(\theta_x) \end{bmatrix}, \mathbf{R_Y} = \begin{bmatrix} \cos(\theta_y) & 0 & \sin(\theta_y) \\ 0 & 1 & 0 \\ -\sin(\theta_y) & 0 & \cos(\theta_y) \end{bmatrix}, \mathbf{R_Z} = \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 \\ 0 & 1 & 0 \\ -\sin(\theta_z) & 0 & \cos(\theta_z) \end{bmatrix}$$

where θ_x, θ_y, and θ_z are the right-hand rotations about the x-, y-, and z- axes, in the order applied by the matrix transformation (X, then Y, then Z). Once this equation is computed, all that must be done in order to determine the actuator lengths is to find the norms of each $\mathbf{L_i}$ vector.

2.1.2 Leap Motion

The Leap Motion controller uses three IR emitters and two cameras to detect a user's hand in 3-D space, and is able to determine the positions of each finger and palm. Using the known normal vector to the palm and its position, this information can be used to move the platform centroid to the desired position and the normal vector matching that of the palm (within the range of the platform). The code used to access the controller/hand data is proprietary and released by Leap Motion as as part of the Leap SDK.

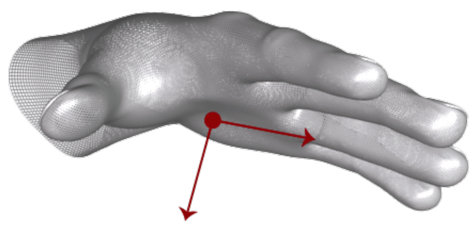

Figure 2: Normal vector to the palm as seen by the Leap Motion controller.

2.2 System Overview

The platform is controlled via an Arduino Due microcontroller, which interfaces with the linear actuators alongside two Multimoto motor controllers and to the host PC over a USB serial connection. Once connected to the host, platform movement can be administered through typing serial input or with the Leap Motion controller using a graphical or command-line interface.

Actuator movements are controlled through a PID feedback system using the PA-14P's potentiometer readings as inputs. The reference parameters are the desired readings/lengths; these are inputted by the user, either manually over serial or from the Leap Motion interfaces using the affine transformation in section 2.1. After running an initial calibration procedure on reset, the platform will run indefinitely in this loop, receiving continuous positional input and adjusting PWM and directional output for each actuator accordingly.

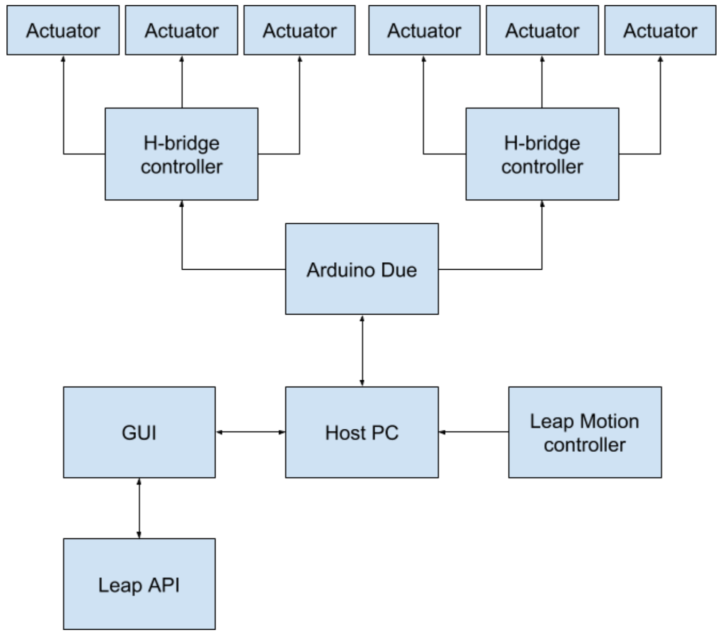

Figure 3: System-level diagram for the platform.

2.3 Methods and Testing

2.3.1 Microcontroller

When the project was received, an Arduino Mega was used to receive commands from the host PC and to set the desired actuator lengths. The previous project group reported issues with serial buffer overflow, which caused the board to freeze and ignore further input.

During the first few months of the project, multiple attempts at optimizing the performance of the Mega were made, including splitting up tasks (e.g. input retrieval, data parsing, feedback control) into separate asynchronous threads, and compressing the command packet size from a maximum of approximately 30 bytes to 12 bytes (by treating each actuator position as a constant 2 byte integer instead of a 4 byte string). Both methods provided significant improvement over the initial implementation, but there was still noticeable input latency and occasional serial buffer overflow.

Other faster microcontroller options were explored – mainly the STM32F411 Nucleo and the Arduino Due (both having ARM processors). Although the Nucleo board has a more extensive range of development tools and libraries within the ARM ecosystem, its analog readings were much noisier than with the Arduino, and it was found to be untenable for positional control without dedicating additional development time towards a more robust feedback and filtering system.

The decision was made to use the Arduino Due, as it did not have the aforementioned issues with its analog inputs and allowed for a much smaller redesign. The wiring and existing Arduino code from the Mega were almost entirely compatible – the only changes being the 3.3V logic (vs. 5V) and the SerialUSB library (vs. Serial) due to the difference in the ARM architecture. After modifying the code to work with the Due, it was found that the platform was much more responsive to Leap Motion input.

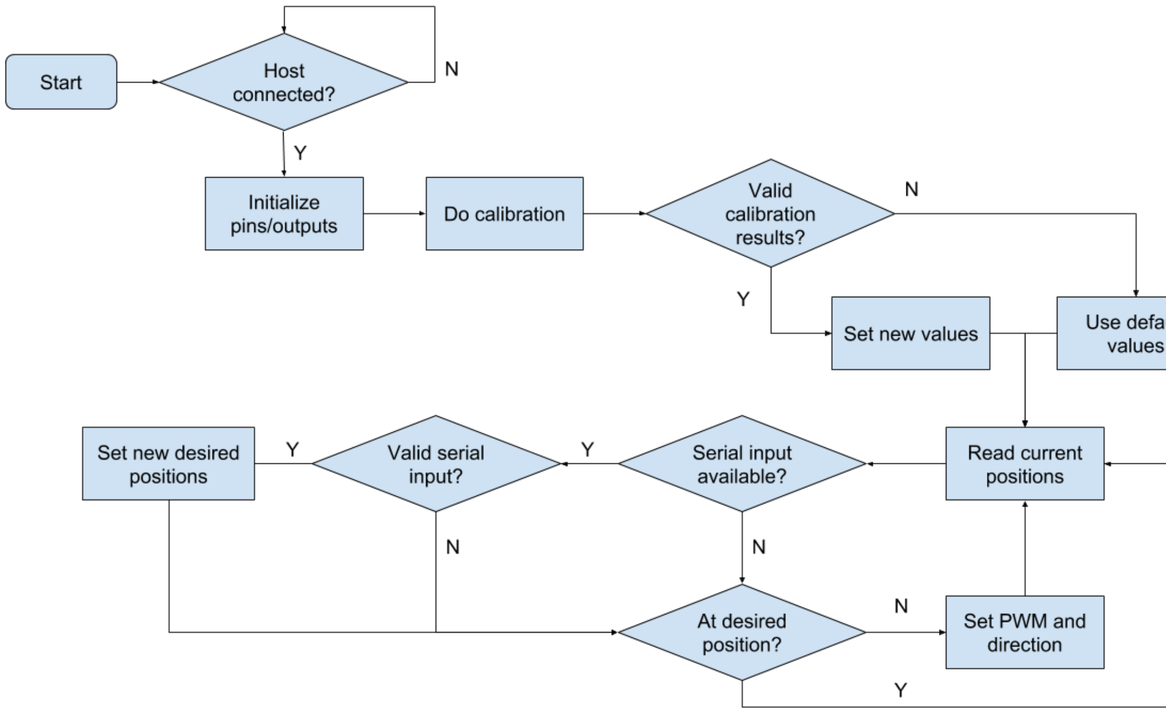

Figure 4: Flow diagram for the Arduino Due.

The software flow for the Arduino is given in Figure 4. After the standard operation of setting pin modes and initial values, the platform initiates a calibration procedure. This is necessary because the potentiometer readings do not span the full range of analog values (0 to 1024), and instead take on an intermediate range (e.g. 188 to 847) that will need to be scaled to the full range to match the input positions. During calibration, the platform takes averaged readings at full extension and retraction and sets those as its internal extrema values. Since this task is relatively brief (less than 10 seconds) and aids in adaptability (i.e. if readings shift over time or an actuator is replaced), it was chosen to occur after every reset. If the readings are invalid or drastically different – such as when the motor controllers aren't powered at reset – the Arduino will defer to a set of default values.

Serial input is given by a string of analog values which correspond to actuator lengths; e.g. `100 200 300 400 500 600`. The actuators are indexed internally in software, but should match the indexing of the platform's mechanical/external connections if wired properly. This is a relatively simple value to parse and process for the Arduino (keeping more intensive calculations on the host side), and allows for easier manual debugging (e.g. testing if an actuator is functional or mapped to the correct index).

While the precision and feedback response of the actuator can be easily characterized, there remains an aspect of subjectivity in user experience when aiming for the best feedback performance. Thus, in order to tune the PID values, the platform first needed to be tested with the Leap Motion and demo functionality completed. From a starting set of empirical values found to perform well in terms of speed of control and responsiveness, the platform was evaluated using a script written in MATLAB, which measured the transient and steady-state response of each actuator at a given desired position. The script was then reiterated and re-run with values slightly adjusted as the results became more stable and consistent. PID values were set for each actuator, as the observed response and noise differed on an individual basis.

2.3.2 Electronics

The complexity of the electronics for this project is fairly low, as the majority of the control is done in software by the Arduino/host PC. However, the redesign of the remaining signal routing and wiring was considered a major project goal due to the concerns of portability and reproducibility with the previous setup. The final printed circuit boards are designed to mate with their respective Arduino/Multimoto boards, which are then connected together using 20-pin ribbon cables. This was done to maximize modularity (i.e. maintain individual boards for separate components) and to simplify mounting and spacing for the enclosure. The boards were fabricated by a third-party vendor, with components later hand-soldered by team members – details of the electronics layout, schematics, and netlist can be found in the appendices.

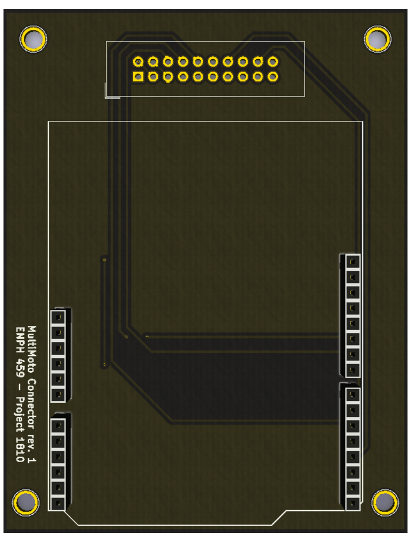

Figure 5: Connector PCB for the Multimoto.

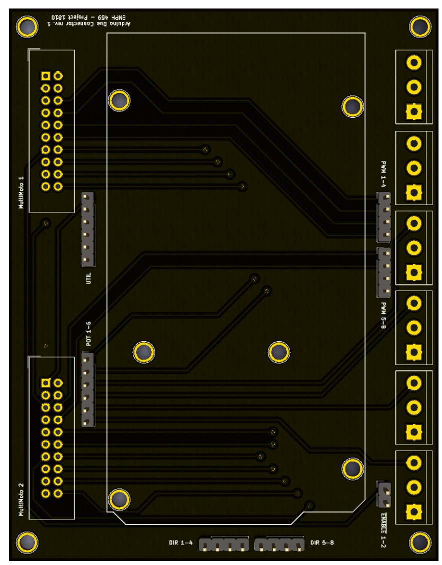

Figure 6: Connector PCB for the Arduino Due.

Since each Multimoto board has a maximum capacity of four motor blocks, the six linear actuators are divided into three per board, with the remaining signal wires (two reference voltages and the potentiometer) connected to the Due PCB's terminal blocks. The Due PCB was designed to support all four of the Multimoto's motor blocks, such that if one became faulty the same motor controller could be used by swapping the wiring to the unused block without having to change the layout in software. The downside to decoupling the hardware from the software in this way is that the actuators are not indexed the same way between these regimes (e.g. the actuator connected to M3 on the Multimoto may be referred to as actuator 1 in code). One will need to keep track of how the actuators are indexed in software by which terminal block they are connected to on the Due

PCB, and confirm that the associated pins are connected to the headers mapping to the actuator's Multimoto block.

To reduce signal noise in the software readings, the potentiometer headers on the PCB are preceded by a low-pass filter with $10\ \mathrm{nF}$ capacitors in parallel with the $10\ \mathrm{k\Omega}$ potentiometer resistances. This improved signal clarity when measuring one actuator with an oscilloscope; however, it was not determined if this would be the optimal value across all actuators and the resistance of the PCB circuit. Depending on if a different actuator model or potentiometer type is used in the future, the capacitance may need to be re-tested and modified.

2.3.3 Electronics Enclosure

At the outset of the project, the plan for the electronics enclosure was to design a small box that would fit all of the circuits and wiring, have all the appropriate connectors, and would sit on the table besides the platform (Figure 7). In this way, it would serve as a "relay" between the controlling laptop (to which the Leap Motion controller is connected) and the actuators of the platform. The initial decision was to 3D print the enclosure, which would allow it to be designed, produced and implemented in a relatively short time frame.

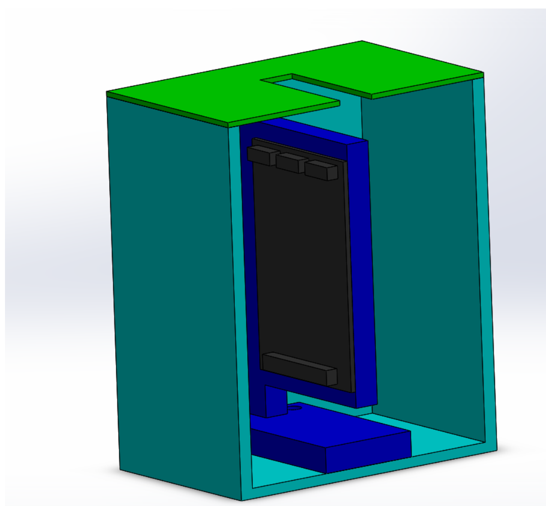

Figure 7: First enclosure design, which was scrapped in favor of a sheet metal enclosure.

However, after several rounds of meetings and consultation with project lab staff, the decision was made to pivot in the main design elements of the enclosure, and increase the scope from a tabletop box to something made from more durable materials (e.g. sheet metal or wood). The enclosure would also be attached to the base of the platform, so as to be unified into a single assembly. This was motivated by the fact that the unified assembly would appear more presentable and professional, and that a more robust material choice would reduce the chance of failure over years of transporting the platform continent-wide.

Folded sheet metal was chosen as the material for the enclosure. To improve heat flow and cable management characteristics, the size of the enclosure was chosen as a rectangle that filled as much of the area beneath the Stewart platform base as possible, with one side installed flush to a long side of the base plate – the power and USB connector are installed on this side to allow to easy connections (Figure 10). The six actuator connectors are placed on the opposing side of the enclosure (Figure 8), which allows the actuator cables to be neatly bundled behind the platform, outside the view of users but easy to access during setup/pack-up.

Considering the entire platform is large and unwieldy, it was decided that the electronics enclosure be suspended from the bottom of the base plate (Figure 11), to allow for easy removal/installation and exposure to the internal electronics. The base requires legs under the base plate to provide consistent support with or without the enclosure suspended. These were made from one inch thick aluminum rod and are topped with adhesive rubber feet, so that they are rigid, and non-slip. There is a clearance of about 3/4" between the bottom of the enclosure and the surface upon which the entire platform assembly rests, which allows some additional space to route cables.

It was raised that heat accumulation within the case could be a concern, due to the power supply and motor drivers. However, it was later realized that the maximum current draw of the power supply is 2.0A. This fact, combined with the knowledge that the motors will be loaded with about 6% of maximum load (Top platform is about 12lb, over 6 actuators, each with maximum load of 35lb), gives confidence that the venting for the power supply fans will adequately cool the electronics in the enclosure.

2.3.4 Control GUI

As mentioned in section 2.3.1, it was established that the host PC would handle the kinematics of the platform, and would send actuator extension lengths over serial to the Arduino. Thus, the message packet format consisted of just six space-separated integers in the range $[0, 1023]$. This was easy to test in the Arduino serial monitor, as it was simple to type six integers, but a more intuitive solution should be provided to the end-user to utilize the platform.

It was decided to use the Qt framework, because it provided cross-platform linear algebra and serial communication libraries, an intuitive IDE for interface development, and an easy-to-use callback ("signals and slots") system for handling UI events. For manual control, the user can use either the sliders or numeric entry boxes to adjust each actuator length. In addition, a logging window is provided for the user to check on the state of the Arduino, and a option to switch over to Leap Motion control was also integrated.

A simpler command-line interface with Leap Motion support was also written in Python; this was developed first as a proof-of-concept implementation for Leap Motion control, and remains a useful tool as it is faster to launch and connect to the Arduino and Leap Motion controller due to internal configuration on start-up.

2.3.5 Demos

After adequate user control of the platform via the Leap Motion control was achieved, a temporary tilt maze was mounted to the top of the platform (Figure 13). This setup was tested at the Engineering Physics high school event. Due to the small ball size and imprecision in control (caused by the overall movement speed), users found it difficult to progress very far in the maze without the

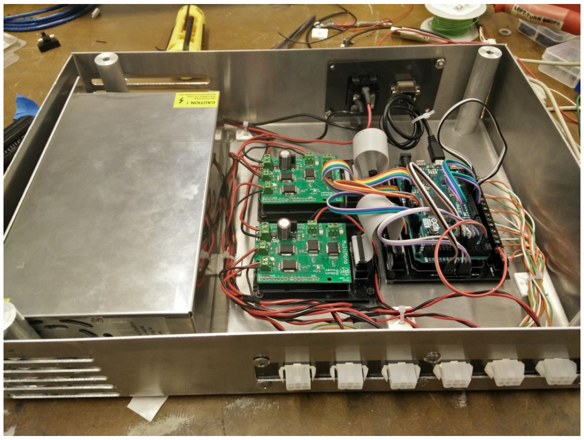

Figure 8: Finished electronics enclosure, rear.

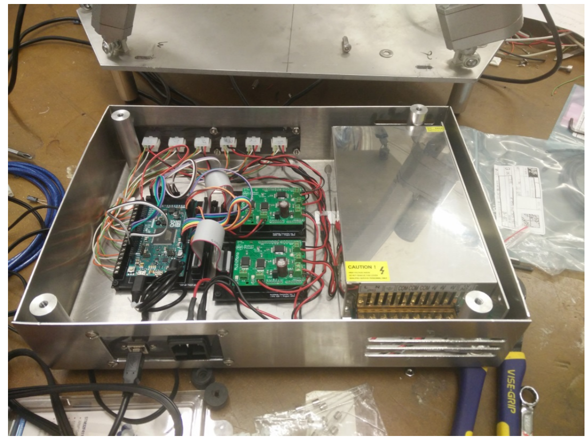

Figure 9: Finished electronics enclosure, side.

Figure 10: Finished electronics enclosure, front.

Figure 11: Electronics enclosure suspended under the platform base (incomplete).

ball falling into one of the many holes; however, despite this it was observed that users were very engaged and enjoyed interacting with the platform. Based on this, it was decided that the primary demo would be a custom-made maze that would be easier to play with, since it is anticipated that users or passersby at a trade show would not spend as much interacting with the platform to warrant a similar level of challenge. This would also be sufficient in demonstrating the functionality of the actuators; further optimization of the game would not be much more productive towards this end.

With this feedback in mind, the new maze was designed to be much wider and easier to navigate. The base was cut out of a wooden sheet, with walls cut out of clear acrylic sheet, so that the demonstration is accessible even to people who are not tall enough to see over the walls (Figure 17). The walls were secured to the base by applying clear epoxy to the joints underneath. Mechanical drawings for the components of this maze are included in Appendix B.

A second proposed demo was to have the platform balance a ball on top, using a resistive touch panel for ball position feedback. Unfortunately there was not enough time to fully explore and complete this demo, and there was difficulty obtaining a breakout board that was compatible with the touch panel connector.

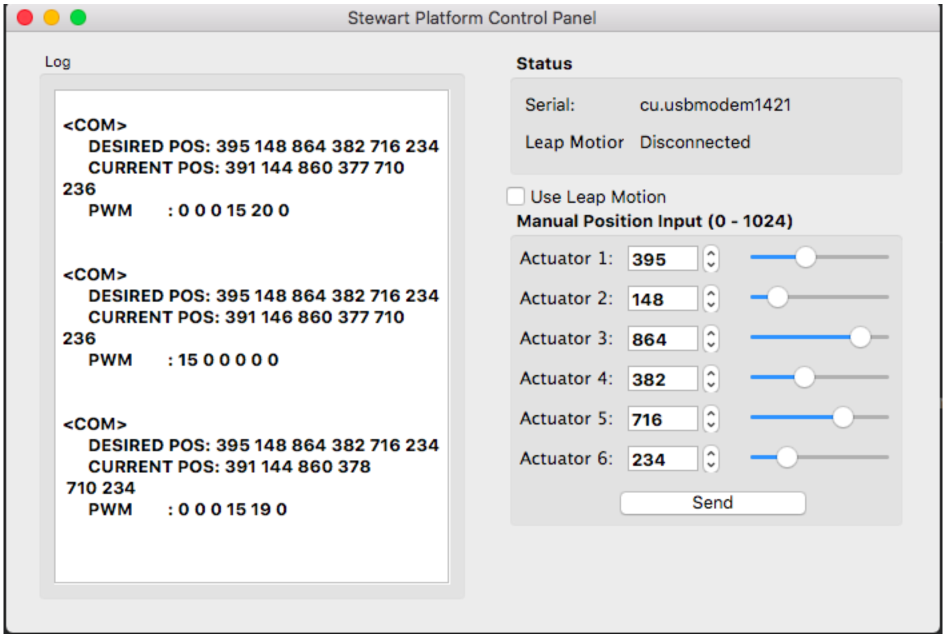

Figure 12: Qt control GUI for the platform.

2.4 Results

The maximum vertical range of motion was measured by setting the actuators at their extreme positions, obtaining the total separation distance between the plates at the extreme points, and subtracting the separation distance at full retraction (found to be approximately 330 mm). These values are found in Table 1.

Range of Motion	**Measured (mm)**
Minimum (1024 1024 0 0 0 0)	-105
Maximum (1024 1024 1024 0 0 1024)	195

Table 1: Comparison of theoretical vertical range of motion to measured values.

From both Leap Motion and manual input, the platform has difficulty in pure horizontal translation (i.e. what was measured in Appendix A). The timing from zero to full extension is approximately 3 seconds (measured from the calibration routine).

Positional performance before/after the PID feedback is shown in the example transient/steady-state response plots (Figures 15 and 16). The pre-PID plot runs on a flat PWM value of 100 (out of a range from 0 to 255) outside of the position threshold, where as the PID plots use the values obtained from the process in section 2.3.1.

As the usability and enjoyability of the demo/tilt maze are not easily quantifiable, this part of our project is analyzed in the following section.

Figure 13: Platform with commercially available tilt maze, used for initial testing and feedback.

2.5 Discussion of Results

Although there were not preliminary calculations on the vertical tilt/range of motion of the platform, it was clear from observation that there was a limit from further movement caused by the ball-and-socket joints at each end – these would either reach their rotational limit or collide with the top plate before full extension. To better understand the limitations, one would need to calculate the theoretical limits using the actuator and platform dimensions in Appendices A & B. It is worth noting that these conditions are only possible through manual input, as the Leap Motion controller is unable to translate the palm orientation at these limits.

The issues with horizontal translation are related to the actuator speed – the actuators further from the direction of translation are assigned a greater speed via the feedback algorithm and therefore has a tendency to tilt. The actuators on opposing sides of the platform are also oriented in opposing horizontal directions (since the top plate is slightly smaller than the bottom plate), so there is a slight movement cancellation when opposing sides extend at similar speeds. The measured vertical speed is as expected, given the stroke length and speed in Table 3.

From the PID plots, there is no significant difference in the positional accuracy of the configurations. Figure 15 once within the threshold (marked by the dashed lines) reduces its PWM to zero, so the noise is purely from the potentiometer reading; Figure 16 also has potential fluctuations caused by proportional/integral corrections within the threshold, but does not exhibit more erratic behaviour.

After the tilt maze was finished and secured to the top plate of the platform, it was tested extensively by the team members, Engineering Physics students, and the project lab staff. In general, users appeared to be engaged with the project. It was demonstrated that the platform responded predictably to rotations of the hand, and users found the control to be sufficient in competently navigating the maze. The latency between control input (moving the palm) and the movement

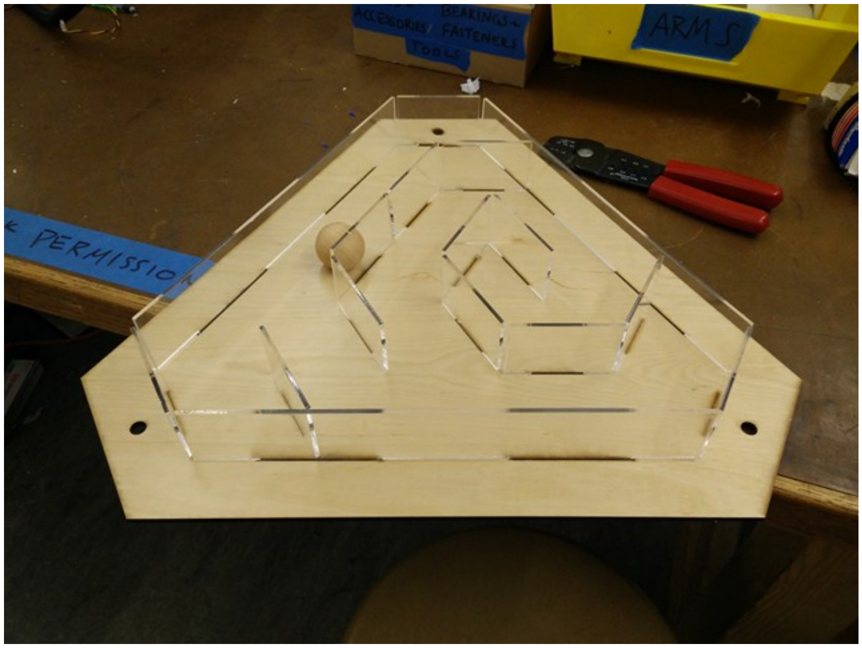

Figure 14: Final design of the tilt maze, with wooden base and acrylic walls.

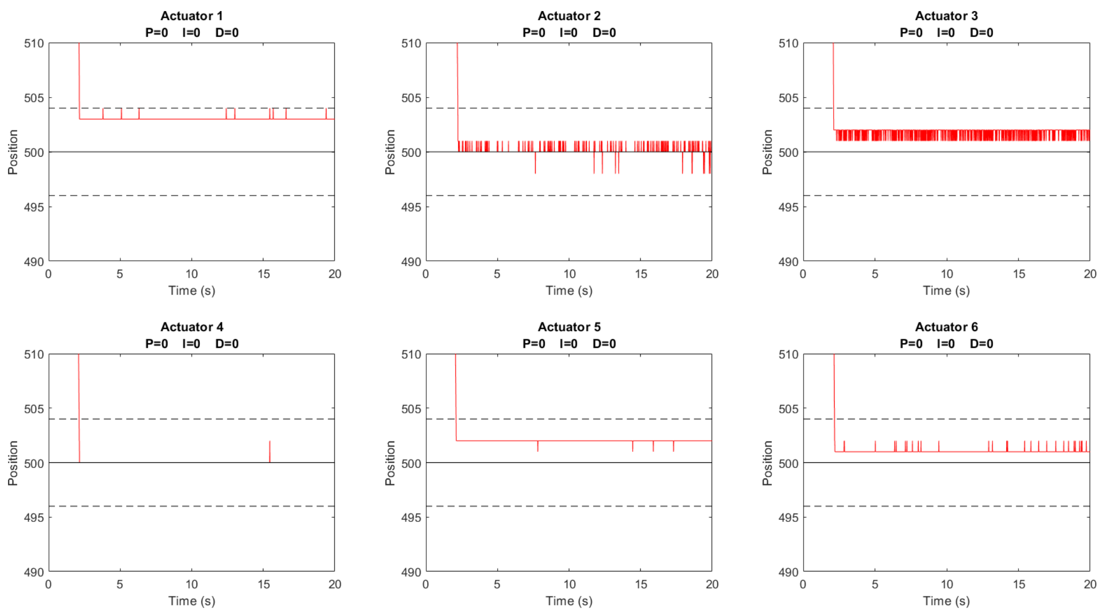

Figure 15: Actuator position response without PID.

of the platform was unnoticeable and did not adversely affect user interaction. The design of the maze was received as professional and well-made; users reported that the wooden base looked clean, and that the clear acrylic walls improved the usability at different table heights.

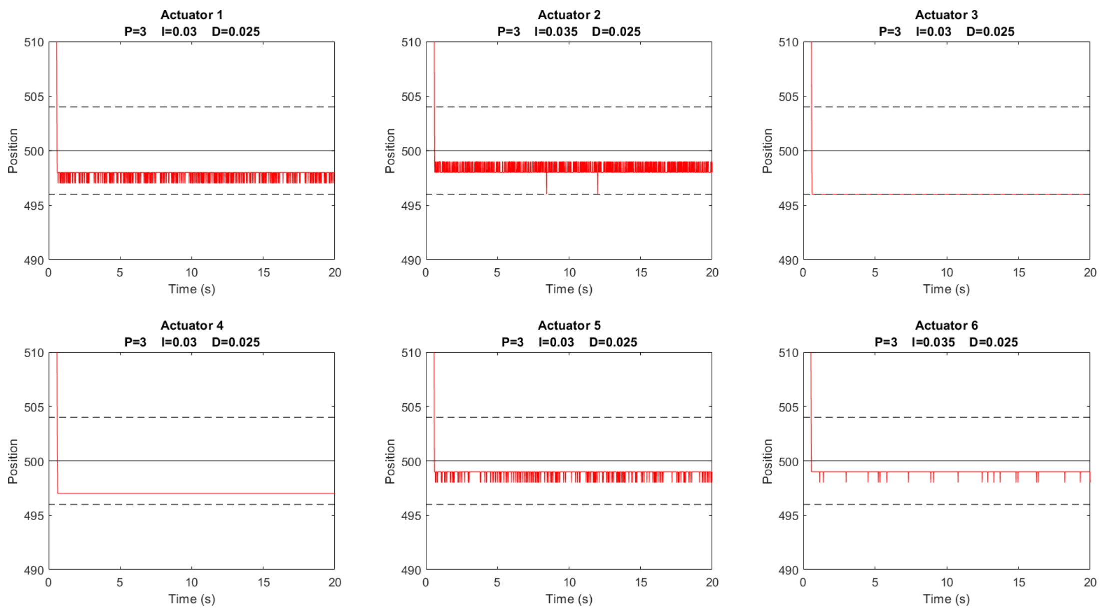

Figure 16: Actuator position response with PID.

3 Conclusions

Though the platform has physical limitations in terms of the translation and tilt, this should not affect the overall performance of the demo – the extreme positions required to reach the vertical height limits are beyond the capability of the Leap Motion controller to capture, and the maze demo emphasizes tilt/rotation over horizontal translation. Given the platform's geometry and these results, the ideal demo archetypes for this platform should specialize in balancing objects and positional stability over free movement.

There is also no discernible difference in precision between feedback methods, so there is little motivation to pursue noise reduction in software. Instead, the controller should focus on the overall smoothness and responsiveness of user input, which was vastly improved but not easily quantified through the results.

As mentioned previously, the tilt maze appears to be very well-suited for the platform. Given the positive user reception and high level of engagement from different audiences, it should act as an eye-catching demonstration at trade-shows and generate interest for Progressive Automations from the general public.

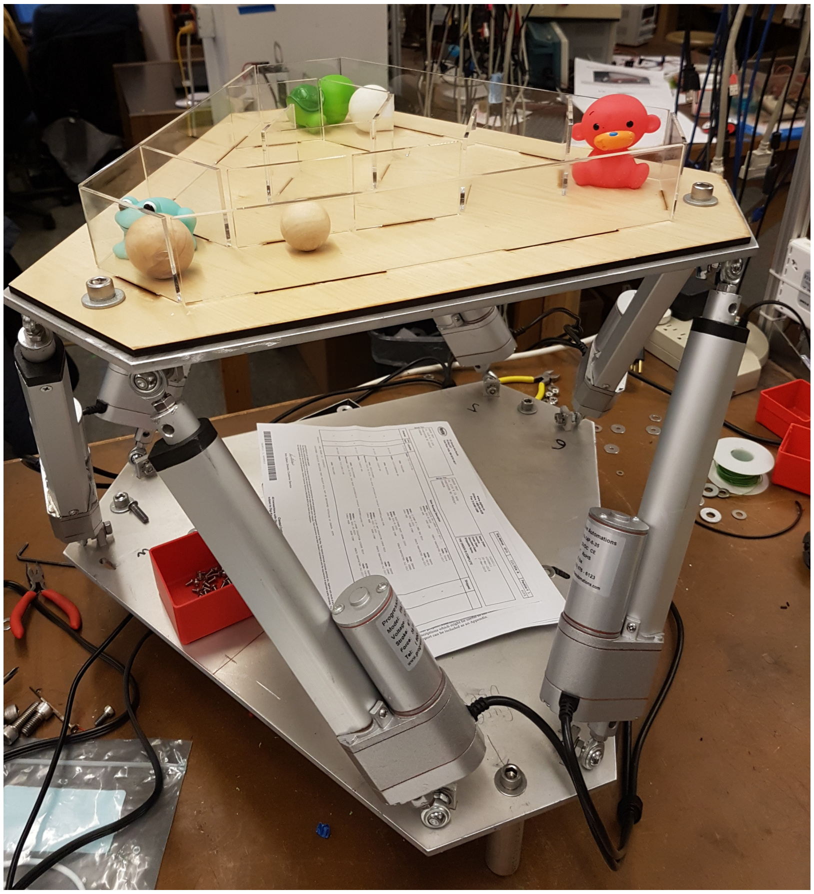

Figure 17: Fully assembled platform with custom maze.

4 Project Deliverables

4.1 List of Deliverables

All project deliverables outlined in the project proposal were met.

- Fully assembled Stewart platform
  - The platform itself was slightly modified from when it was first received, including:
    - holes drilled on the top plate for mounting the demo(s);
    - holes drilled on the bottom plate for mounting the electronics enclosure and feet; and
    - feet attached to the bottom plate.
  - A maze was assembled and mounted on the top plate via the aforementioned holes.
- Electronics enclosure (circuits, controllers, and power supply)
  - This section of the project fell behind the proposed schedule due to redesign, but has been completed and mounted to the bottom of the platform.

- Complete software package (source code, compiled binaries, and APIs/libraries)
  - All the source code is available on GitHub[2] (including the Qt GUI, Arduino program, and other scripts used for development).
- External control method(s) for the platform
  - This is the same Leap Motion controller given at the beginning of the project.
- Documentation for the platform (operation instructions, technical capabilities, mechanical drawings, schematics, etc.):
  - All related project files (schematics, drawings) are located in their respective subfolders in the GitHub repository.
  - Documentation is also located in the repository as markdown files.

4.2 Financial Summary

Table 2 is a list of all the items purchased for the project. Most of the mechanical hardware was procured from the Engineering Physics project lab stores.

4.3 Ongoing Commitments

Beyond the current state of the project, the team members are capable of providing technical support through software and documentation updates. It is unlikely that team members will be help physically with the platform once handed-off, but can aid in setting up the software and dependencies to control the platform on the sponsor's host PC. Team members will be able to support the sponsor through the summer of 2018 up to September.

[2] `https://github.com/henrymliu/StewartPlatform`

Vendor	Purchaser	Item	Quantity	Unit Cost	Total Cost
Digikey	Project Lab	Terminal block (6 holes) (Molex 0398800306)	10	$1.93	$19.30
Digikey	Project Lab	Terminal block (3 holes) (Molex 0398800303)	25	$0.89	$22.23
Digikey	Project Lab	10nF SMD caps (Kemet C0805C103K5RACTU)	20	$0.05	$1.08
Digikey	Project Lab	Power connector (Qualtek 723W-X2/02)	1	$2.12	$2.12
Digikey	Project Lab	USB connector (Amphenol MUSB-D511-00)	1	$14.01	$14.01
Digikey	Project Lab	USB A male to micro B male cable (Qualtek 3025010-03)	1	$3.45	$3.45
Digikey	Project Lab	Molex connector kit (Molex 76650-0076)	5	$1.73	$6.73
Digikey	Project Lab	6.3A fuses (Bel Amp Inc. 5ST 6.3-R)	10	$.208	$2.08
McMaster	Project Lab	Stainless Steel Nylon-Insert Locknut, 1/4"-28 Thread Size (91831A120) (Pack of 50)	1	$5.72	$5.72
McMaster	Project Lab	Stainless Steel Hex Nut, 1/4"-28 Thread Size (91845A105) (Pack of 100)	1	$4.47	$4.47
McMaster	Project Lab	Ball Joint Linkage, 1/4"-28 Thread, Right-Hand Shank, Right-Hand Ball Stud (60645K221)	6	$4.91	$29.46
PHAS Store	Project Lab	Al rod, 12” length of 1”⌀	.94lb	$8/lb	$7.54
PHAS Store	Project Lab	Al rod, 15” length of 3/4”⌀	.66lb	$8/lb	$5.30
PHAS Store	Project Lab	Al sheet, 365.5 in^2 rectangle, .060in thick	2.19lb	$11/lb	$24.12
PHAS Store	Project Lab	Electrical Wire, 3ft	5	$.22/ft	$3.30
Newark	Sponsor	STM32F411 Nucleo	1	$15.57	$15.57
Amazon	Sponsor	Arduino Due	1	$59.95	$59.95
PCBWay	Don	Multimoto PCB	5	$.64	$3.20
PCBWay	Don	Arduino Due PCB	5	$6.98	$34.89
Amazon	Don	20-pin JTAG IDC sockets (DC3-20P) (Pack of 20)	1	$5.17	$5.17
Amazon	Don	20-pin IDC ribbon cables (FC20P) (Pack of 5)	1	$7.09	$7.09
		Sum (excluding sponsor-purchased items):			$201.26
		Sum:			$276.78

Table 2: Components purchased for the project.

5 Recommendations

1. **Purchase a protective case for the electronics.**
   - A protective case is recommended for the electronics enclosure for transportation, as it would likely be subject to rough handling while being transported (e.g. on an airplane).
   - Note that the dimensions of enclosure are fairly large so that high-end options such as a Pelican case may be cost-prohibitive; in this case cushioning with styrofoam cutouts/blocks would likely be sufficient if placed in a larger box/enclosure for transportation.
2. **Use more durable wiring for the Arduino Due.**
   - The current jumper cables from the Arduino to its breakout board are too long, and it is probable for them to accidentally disconnect during transportation or re-wiring.
   - Shorter cables that are more rugged (difficult to pull from the pin headers) should be investigated and used instead.
3. **Improve the Qt GUI for better presentation.**
   - The current GUI is rudimentary, especially in terms of logging – reading serial text on Windows is particularly painful due to the parsing of line endings (appears to perform better on Mac). Improving the `readSerialData()` functionality to parse more selectively would greatly improve the reading of platform data.
   - Adding an 'About' page under the menu bar may be considered if the software is distributed or finds further use.

6 Appendices

A Stewart Platform Parameters

The specifications for the PA-14P linear actuator are as follows:

Stroke Length	Minimum Length	Maximum Length	Speed (unloaded)
6 in	11.51 in	17.51 in	2 in/s
152 mm	292 mm	445 mm	50.8 mm/s

Table 3: Basic dimensional and performance values for each linear actuator.

Using the values in Table 3 and the dimensions of the plates in Appendix B, we can obtain theoretical maximum performance values for the platform.

Range of Motion (horizontally; with 10" separation between plates; no tilt)	90 mm, in any direction
Speed (horizontally; with 10" separation between plates; no tilt)	71 mm/s, in any direction
Speed (vertically; with 10" separation between plates)	75 mm/s, in any direction

Table 4: Estimated performance values based off of the actuator and platform specifications.

B Mechanical Drawings

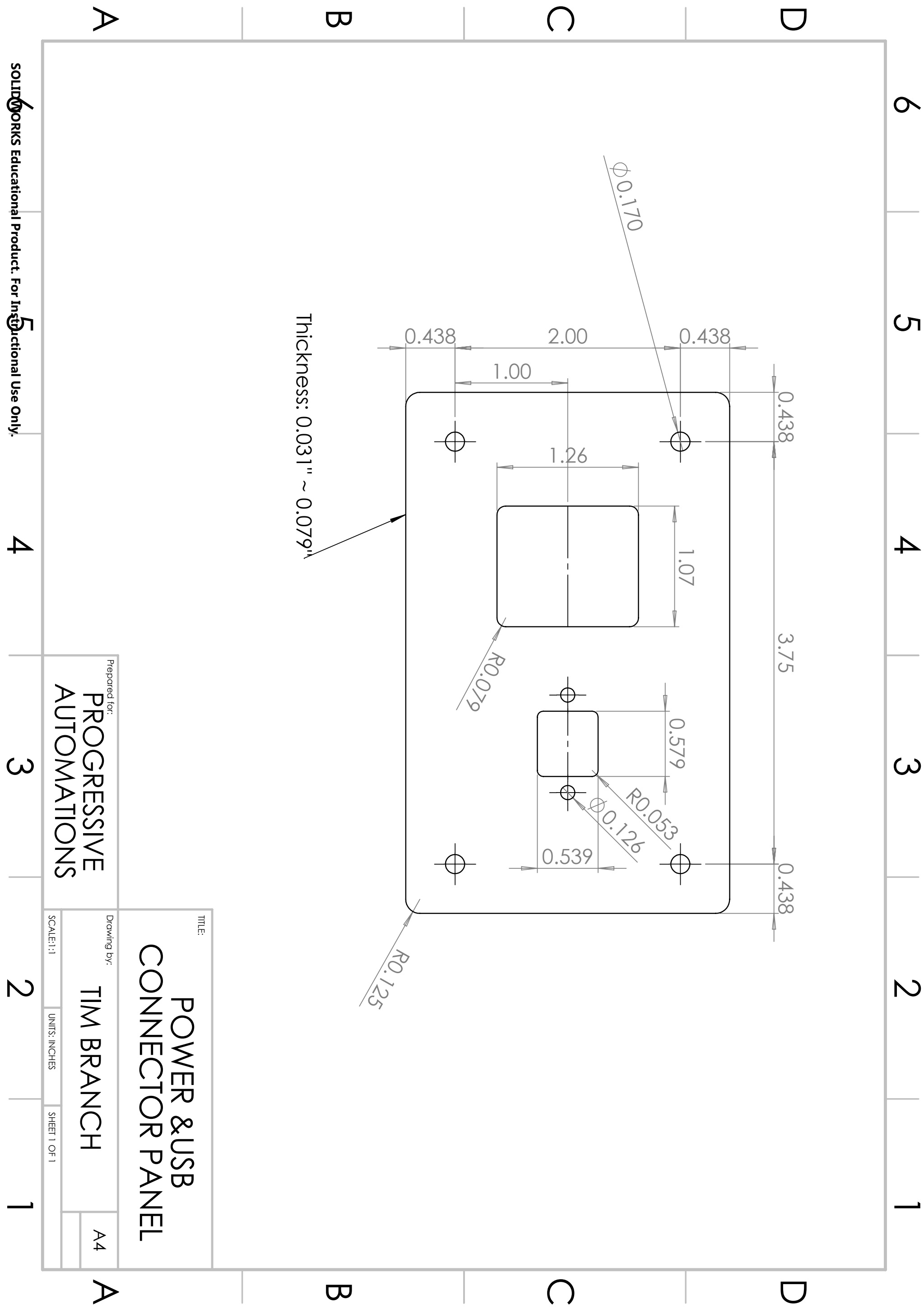

Figure 18: Drawing of the power & USB connector panel.

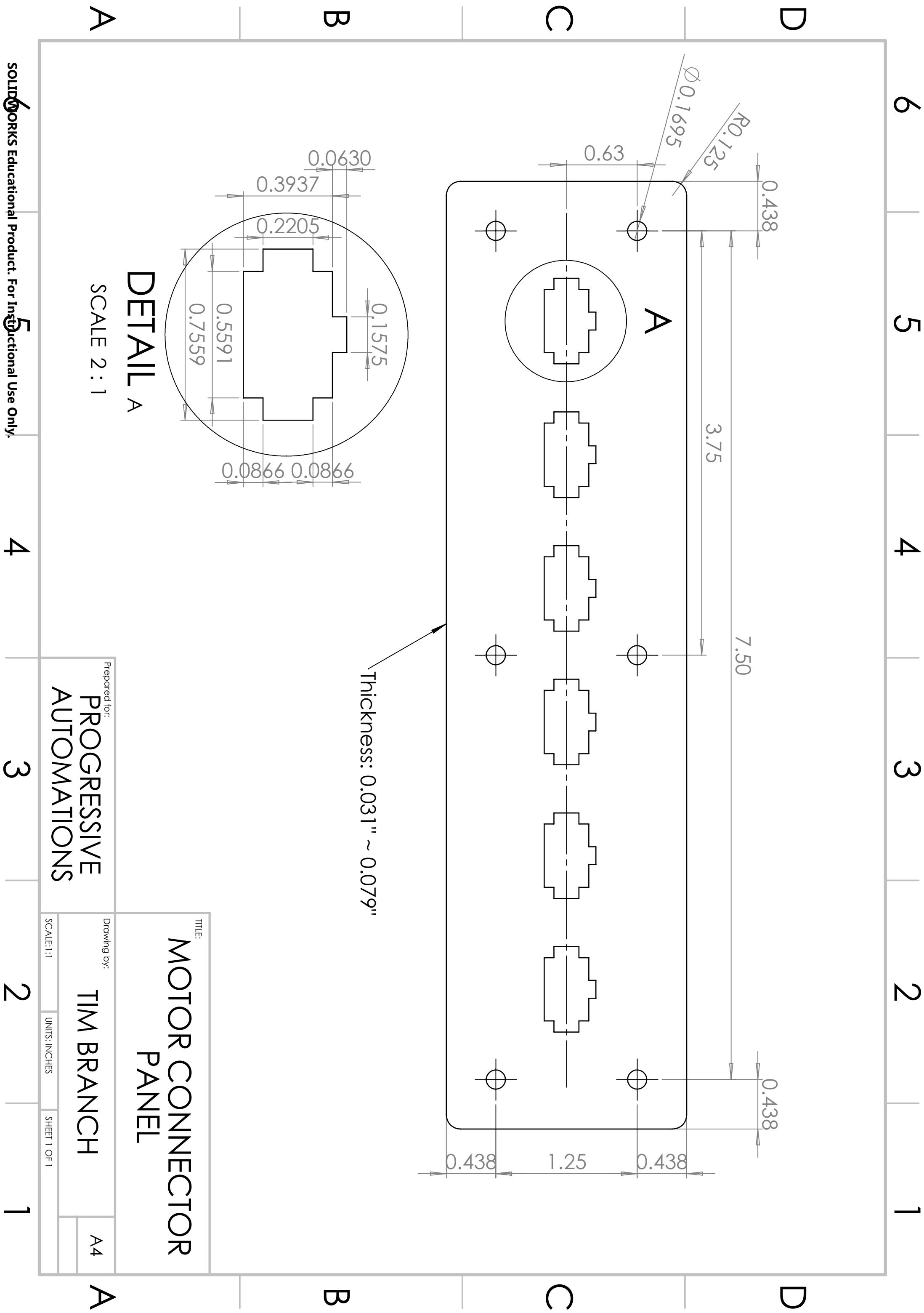

Figure 19: Drawing of the motor connector panel.

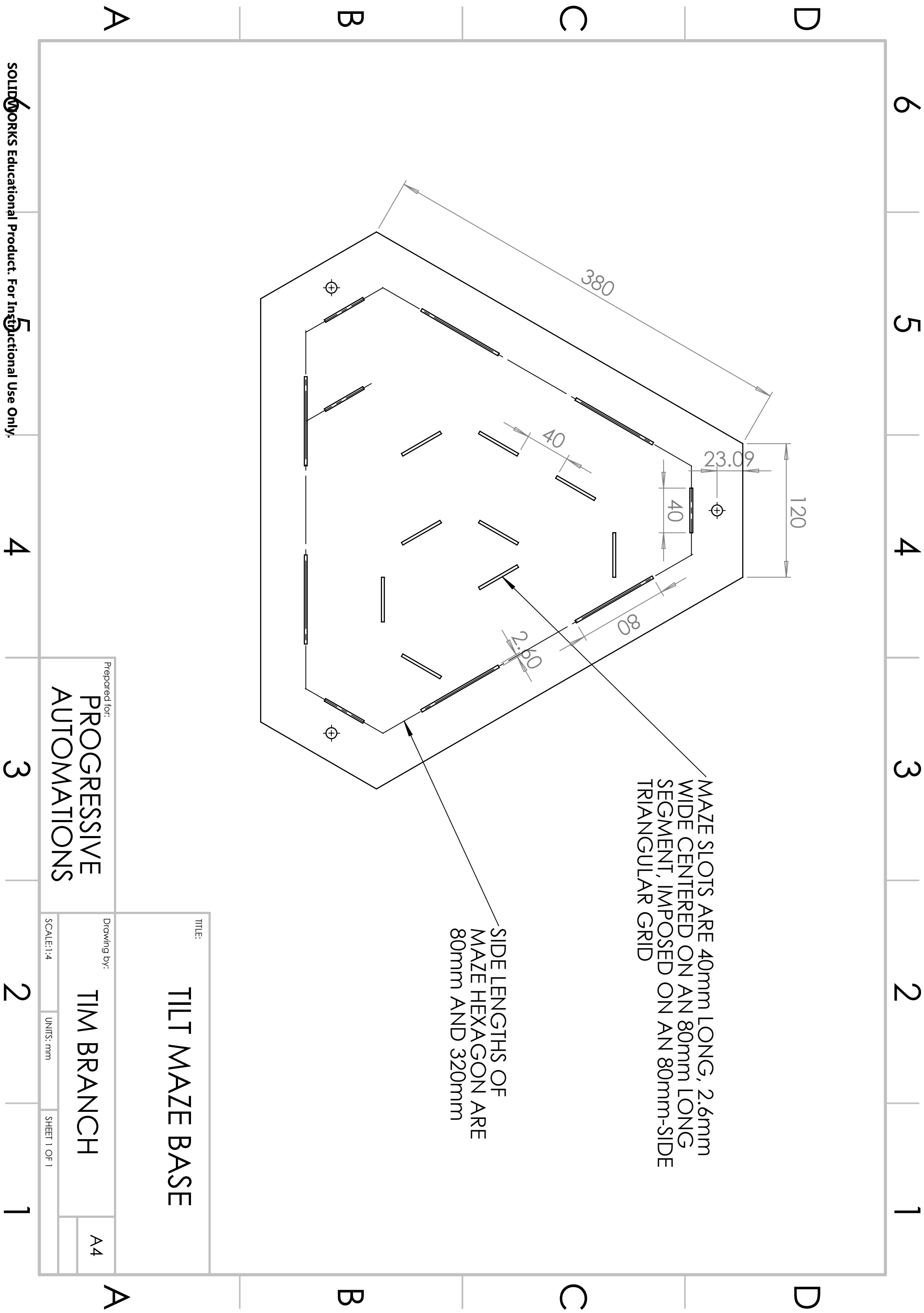

Figure 20: Drawing of the tilt maze base.

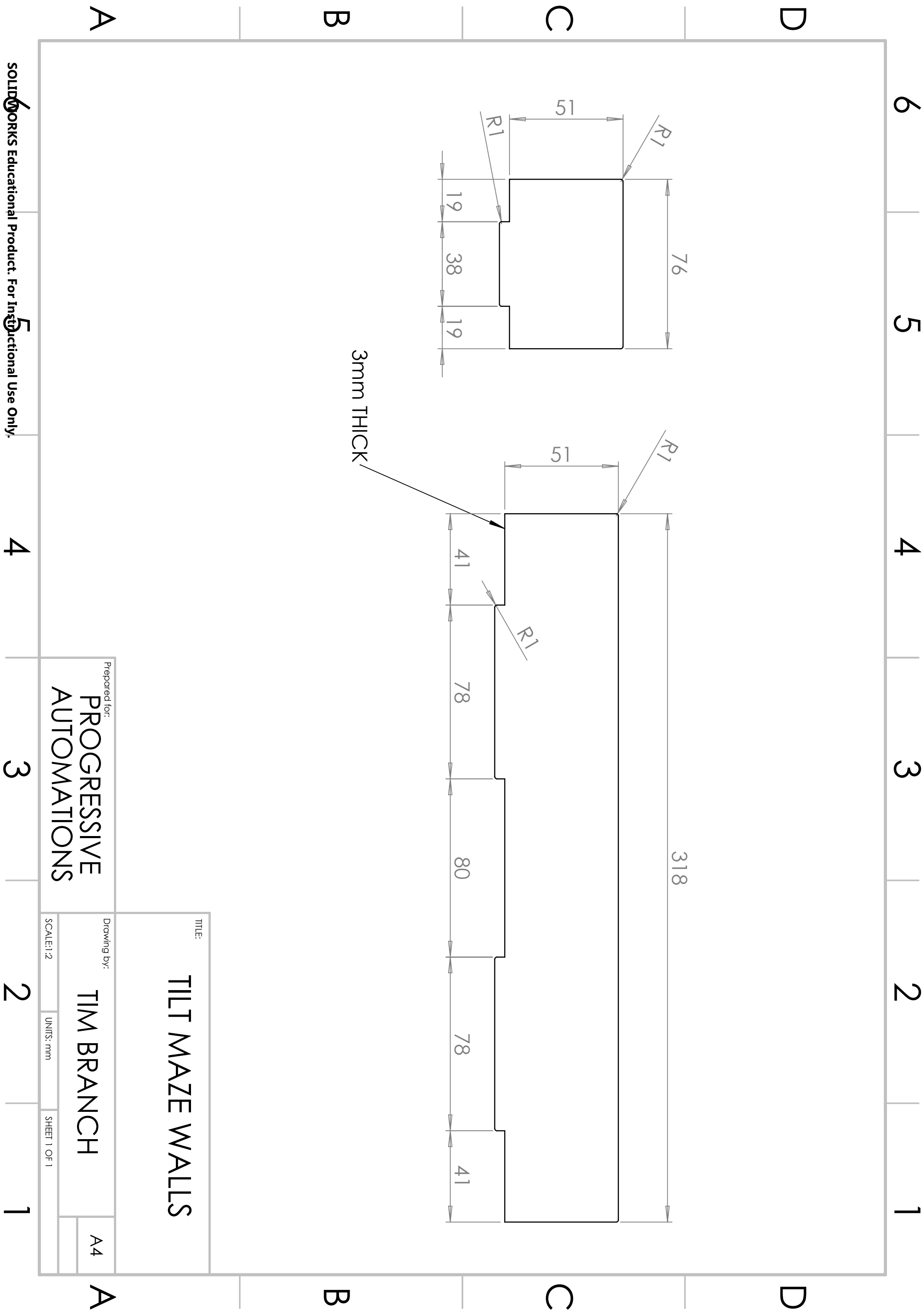

Figure 21: Drawing of the tilt maze walls.

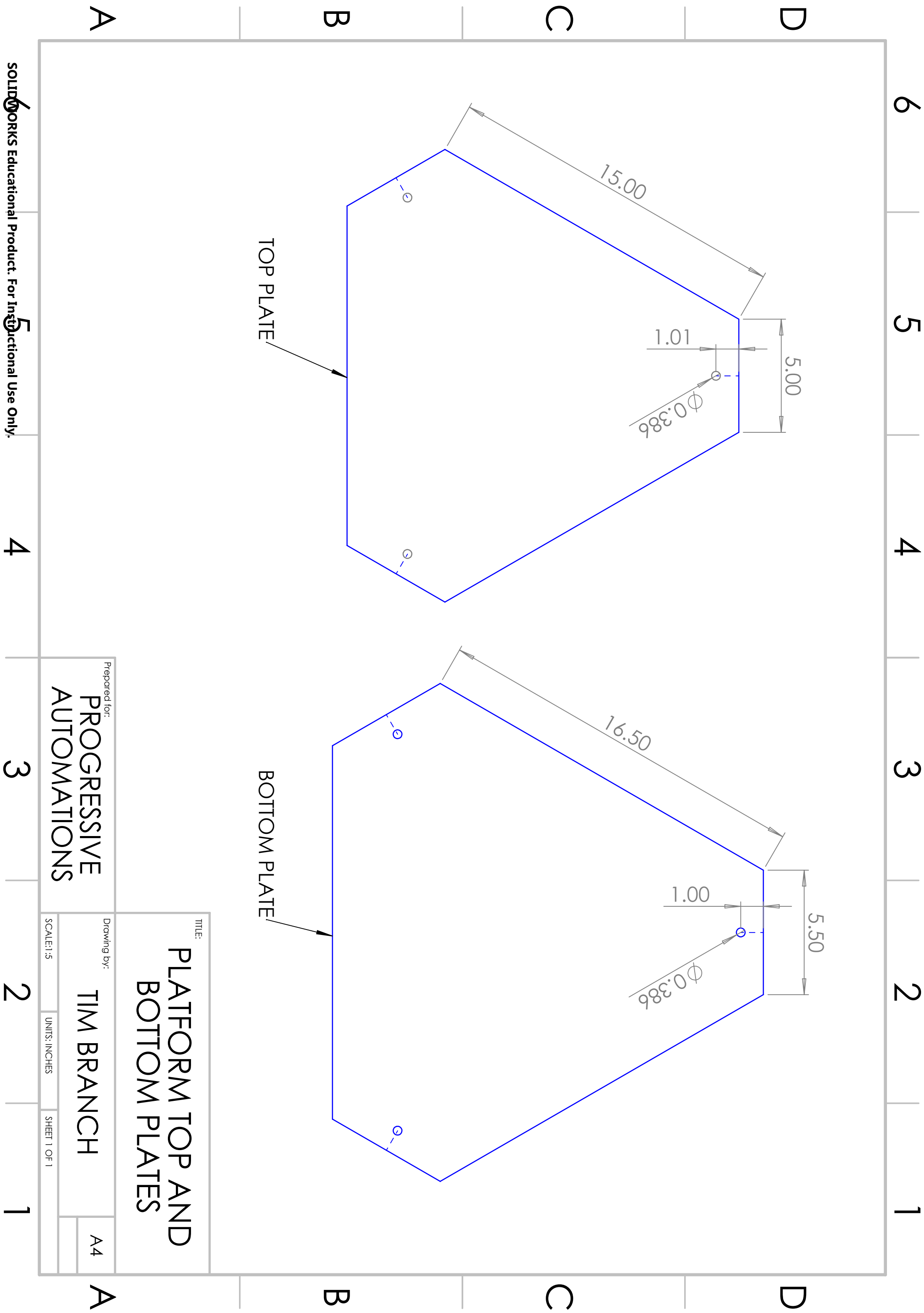

Figure 22: Drawing of the top and bottom Stewart platform plates.

C Electronics Schematics

This appendix contains an overview of the electronics layout and schematics of the connections within the PCBs. The information here should provide sufficient knowledge to wire the boards and components (further detail and design files can be found in the project repository).

The wiring/connections for the Due PCB headers (Figure 24; as of June 28, 2018) are in Table 5. The UTIL pin headers (RESET, +3.3V, +5V, GND, GND, Vin) are connected to the identically-named Arduino pins and are omitted from this list.

Header	**Pin**
POT_1	A6
POT_2	A7
POT_3	A8
POT_4	A9
POT_5	A10
POT_6	A11
DIR_1	41
DIR_2	39
DIR_3	37
DIR_5	35
DIR_6	33
DIR_7	31
PWM_1	13
PWM_2	12
PWM_3	11
PWM_5	10
PWM_6	9
PWM_7	8
$\overline{\text{ENABLE}}$_1	24
$\overline{\text{ENABLE}}$_2	25

Table 5: Pin layouts for the Arduino Due.

NOTE: The numbering for the actuators on the PCBs is indexed according to the Multimoto interface (1 to 4 for Multimoto 1, 5 to 8 for Multimoto 2) which does not necessary align with the numbering of the actuators for the platform (as discussed in section 2.3.2).

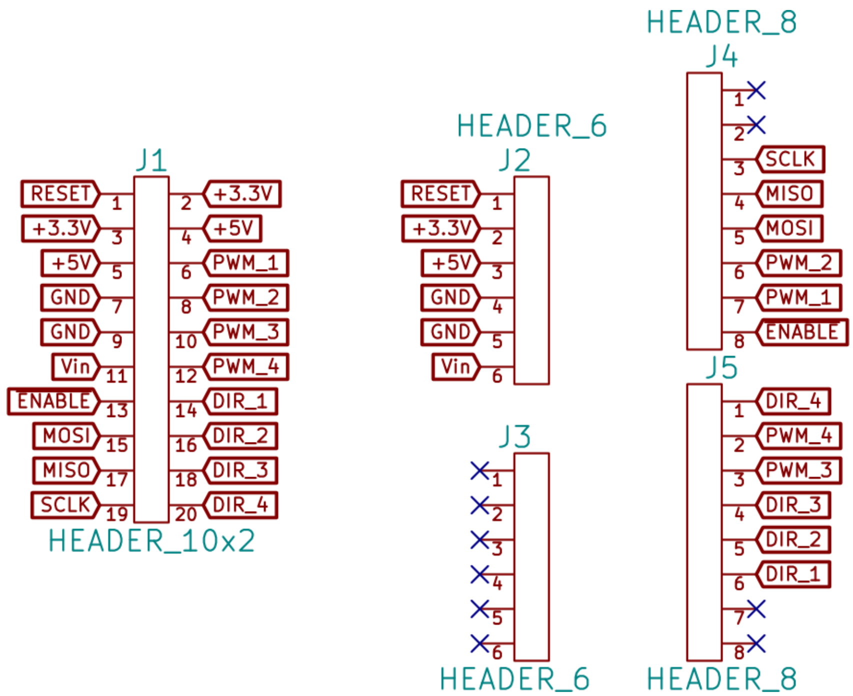

Figure 23: Pin/header layout for the Multimoto connector PCB.

<u>NOTE:</u> The SPI pins on the Multimoto (MOSI, MISO, SCLK) are shown here but are left omitted/unused on the Due PCB as they were found redundant during the hand-off with the previous project group while the boards were being developed.

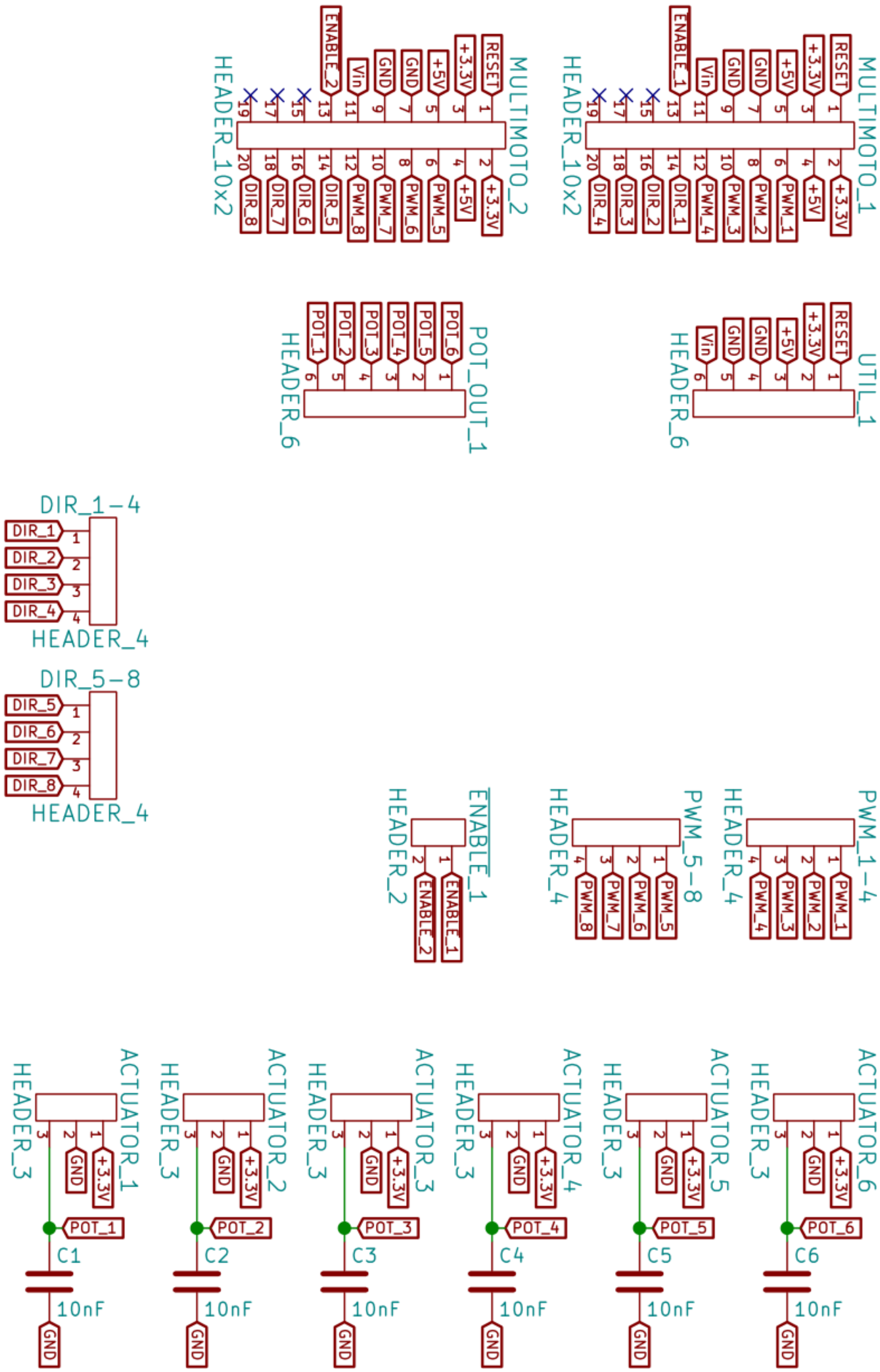

Figure 24: Pin/header layout for the Arduino connector PCB.

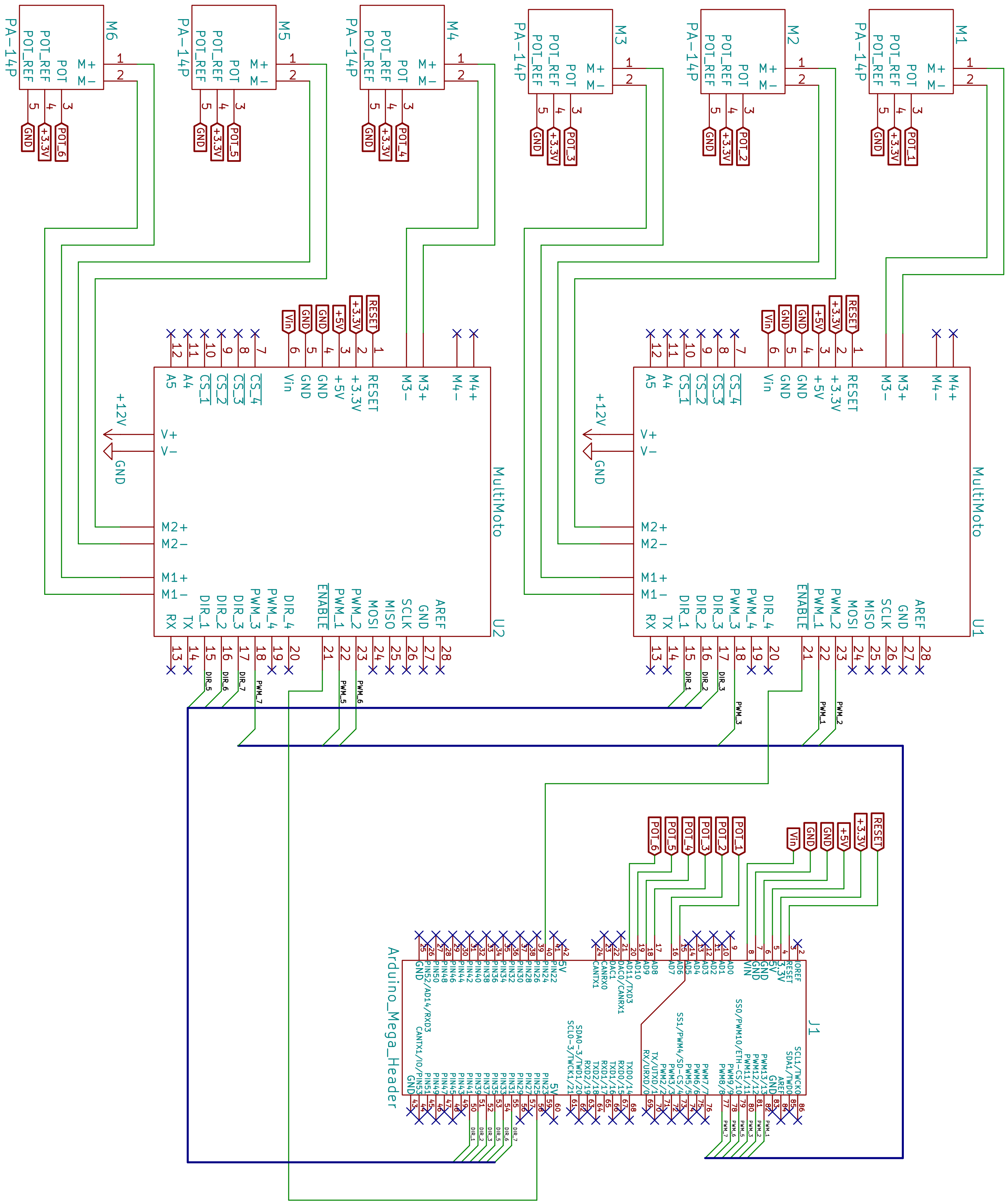

Figure 25: Overall electronics layout.

<u>NOTE:</u> An Arduino Mega header is used for the Due since the pin layout is identical.

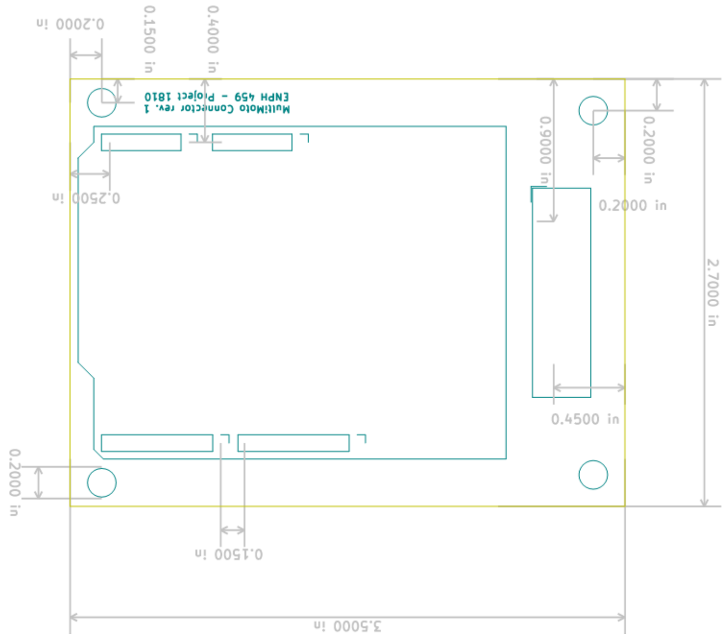

Figure 26: Dimensions for the Multimoto connector PCB.

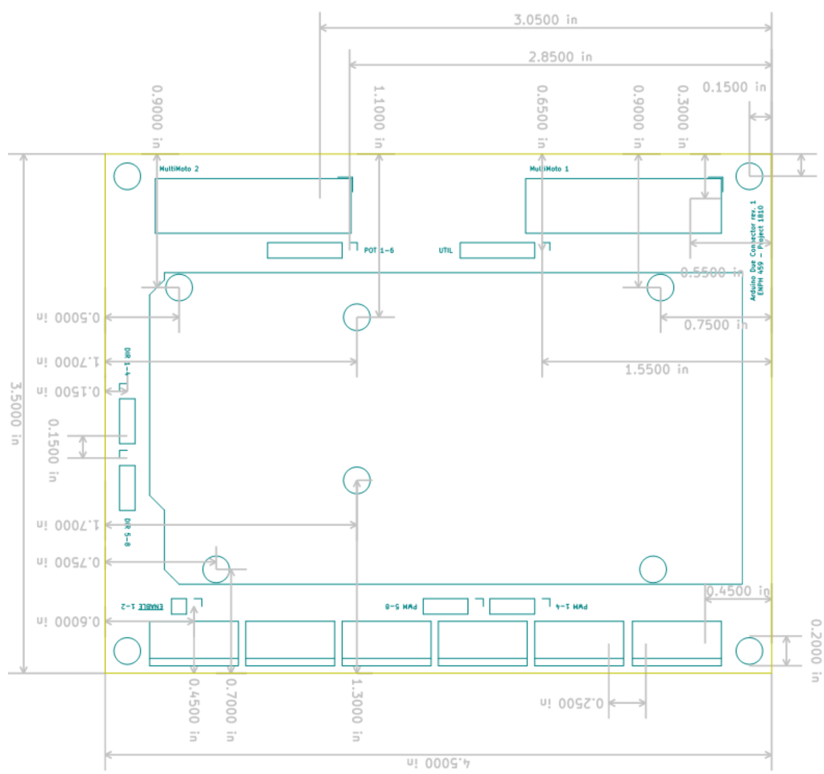

Figure 27: Dimensions for the Arduino connector PCB.

D Bill of Materials

Vendor	Item	Quantity
Digikey	Terminal block, 5.08mm pitch (3 holes) (Molex 0398800303)	6
Digikey	10nF capacitors, 0805 SMD (Kemet C0805C103K5RACTU)	6
Digikey	Power connector (Qualtek 723W-X2/02)	1
Digikey	USB connector (Amphenol MUSB-D511-00)	1
Digikey	USB A male to micro B male cable (Qualtek 3025010-03)	1
Digikey	Molex connector kit (Molex 76650-0076)	6
Digikey	6.3A fuses (Bel Amp Inc. 5ST 6.3-R)	2
PHAS Store	Al rod, 3.5" length of 1"⌀	3
PHAS Store	Al rod, 13.2" length of $3/4$"⌀	4
PHAS Store	Al sheet, 21.5in $*$ 17in rectangle, .060in thick	1
Amazon	Arduino Due	1
PCBWay	Multimoto PCB (schematic in repository)	5
PCBWay	Arduino Due PCB (schematic in Repository)	5
Amazon	20-pin JTAG IDC sockets (DC3-20P) (Pack of 20)	1
Amazon	20-pin IDC ribbon cables (FC20P) (Pack of 5)	1
McMaster	Stainless Steel Hex Nut, 1/4"-28 Thread Size (91845A105)	24
McMaster	Ball Joint Linkage, 1/4"-28 Thread, Right-Hand Shank, Right-Hand Ball Stud (60645K221)	6
Project Lab	Stainless Steel Button Head Bolt, 3/8-16 Thread Size, 3/4" Long	8
Project Lab	Stainless Steel Washer, 1/4" ID	38
Project Lab	Stainless Steel Lock Washer, 1/4" ID	16
Project Lab	Stainless Steel Socket Head Bolt, 1/4-20 Thread Size, 3/4" Long	3
Project Lab	Stainless Steel Socket Head Bolt, 1/4-20 Thread Size, 1 1/4" Long	3
Project Lab	Stainless Steel Washer, 1/4" ID	9
Project Lab	Stainless Steel Hex Nut, 3/8"-16 Thread Size	3

Project Lab	Stainless Steel Button Head Bolt, 8-32 Thread Size, 3/8" Long	10
Project Lab	Stainless Steel Nylon-Insert Locknut, 8-32 Thread Size	10
Project Lab	Stainless Steel Bolt, M3 Thread, 1cm Long	4
Project Lab	Stainless Steel Button Head Bolt, M3 Thread, 1cm Long	4
Project Lab	Stainless Steel Button Head Bolt, M3 Thread, 6mm Long	12
Project Lab	Plastic Button Standoff, M3 Thread, 6mm Wide	12
Project Lab	Plastic Button Hex Bolt, M3 Thread	12
Project Lab	Male pin header, 2.54mm pitch	30
Project Lab	Female pin header, 2.54mm pitch	28

Table 6: Bill of materials.

7 References

[1] D. Stewart. *A Platform with Six Degrees of Freedom*. Proceedings of the Institution of Mechanical Engineers, Vol 180, Issue 1, pp. 371-386, 1965, doi: 10.1243/PIME_PROC_1965_180_029_02.

[2] *Creating a Stewart Platform Model Using SimMechanics*. https://www.mathworks.com/company/newsletters/articles/creating-a-stewart-platform-model-using-simmechanics.html.

[3] Lionel Birglen. *Haptic Devices Based on Parallel Mechanisms. State of the Art*. Jan. 15, 2003. http://www.parallemic.org/Reviews/Review003.html.

[4] Brandon N. Dick, Christopher Oesch, and Timothy W. Rupp. *Linear Actuator System for the NASA Docking System*.

[5] Selçuk Kizir, and Zafer Bingul. *Position Control and Trajectory Tracking of the Stewart Platform*. Serial and Parallel Robot Manipulators - Kinematics, Dynamics, Control and Optimization, Dr. Serdar Kucuk (ed.), InTech, Mar. 30, 2012, doi: 10.5772/35569. http://esmats.eu/esmatspapers/pastpapers/pdfs/2017/dick.pdf.