Stepper and Control Report

Jesse Taylor 16042846 Josh Baynes 15124393

Contents

1 Introduction

This report will summarise the work completed to complete the objectives of the assignment as much as possible. The report will provide a description of the high level specifications of the system, as well as the inspirations for the design of this solution. It will then describe the design of the mechanical solution, the PCB, and the design of the code for both the chosen microcontroller and the PC controlling the solution. While a physical mechanical solution cannot be completed, the theoretical solution presented is evaluated for potential weak points and additional features that could be added.

2 Specifications

For this mechanical solution, we want a solution that looks clean and professional. The concept for this design is greatly inspired by the SnapMaker2.0 project, and its modular linear actuators.
For the solution to look professional, there should be minimal external and internal wiring, as well as a clean, enclosed exterior and casing.
We also want the solution to perform professionally as well, with a high level of control, and very little noise. To achieve this, TMC2130 bipolar stepper motor drivers will be used for their massive list of features. Some of the features we aim to use in this solution are;
Over current detection for auto calibration
Step loss detection
Microstep control
Microstep interpolation
StealthChop, for silent movement.

For the microcontroller, it was chosen to use the AT-Mega 32u4, for its ease of interfacing with USB, the chosen communication method. USB was chosen as it can provide a high speed communication protocol, as well as onwire power for the microchips in the solution. The AT-Mega 32u4 can interface directly with USB, using the D+ and D- signals, removing the need for an external FTDI or similar chip.

The selection of these components, specifically the driver, also allows the solution to have high repeatably, further improving the professional nature of the solution.

3 Mechanical Design

3.1 Design Process

The design process that was used involved obtaining the dimensions for the different components that were to be embedded within the design. These components included, the pcb and the stepper motor. Once these dimensions were obtained current 3d printing systems were then researched and it was decided that a system like that of the snap maker 3d printer.
Once the research phase was complete the idea generation phase was entered. This consisted of sketching what the system would look like and how the different mechanisms and components would assemble. From the sketching phase it was apparent that the most suitable way to assemble was with the use of bolts and threads to ensure the structure of the system is rigid.
Commencing the research and sketching phase the computer aided design phase was entered. This involved taking the estimated sketches and designing each component of the system via solid works. This ensured that the assembly of the mechanical system was feasible and ensured that accurate dimension of each part of the mechanical system could be defined.
Following the CAD design a motion study of the system was carried out with solid works to simulate the behaviour of moving mechanisms. These included the movement of the guide along the lead screw and the rotation of the lead screw. A video of the simulation was submitted with this report.

3.2 Methodology

3.2.1 Enclosure

The first component of the mechanical system to be defined was the enclosure. This was because the enclosure's dimensions were constrained by both the stepper dimensions and the size of the pcb due to these components are to be placed within this enclosure. The enclosure also needed to allow for a lead screw to be mounted to the stepper motor. Shown below is the schematic for the enclosure which includes: 4 x 4mm cut extrudes for the mounting of the stepper motor 4 x 4mm cut extrudes as slots for long bolts 1 x 23mm cut extrude for the stepper shaft and lead screw attachment 1x70mm air vents have also incorporated to allow for heat dissipation

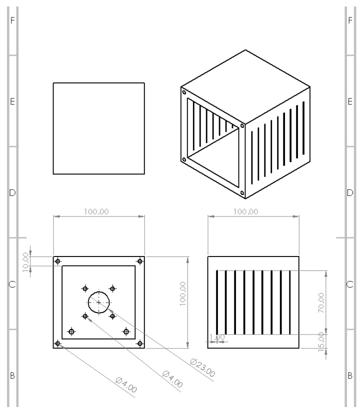


Figure 1: Enclosure Technical Drawing

3.2.2 Lead Screw

The next component was the lead screw, this was chosen due to belt drives being less accurate and slip is often involved with belts. The lead screw consisted of a pitch of 3mm that is to every 1 revolution the guide will move 3mm. Many lead screws can be bought straight from the shelf however for modelling purposes this was designed via solid works that can mount directly to the stepper motor. The schematic shown below includes grub screw slots for mounting. The thread stops prior to the end of the shaft so that a bearing can be pressed onto the end of it

3.2.3 End plate

Following the design of the lead screw the end plate was the next component to be designed within the cad program. The purpose of this was so the bearing which is pressed onto the lead screw could be pressed into the endplate as well. The end plate also allowed holes for the screws so the it could be fastened to the cover of the system and threaded holes for the guide rails. In summary the end plate serves the purpose for supporting the lead screw while it rotates and allows mounting holes to improve the rigidity of the system. Shown below is the schematic for the end plate.

3.2.4 Guide Rails

The guide rails were made so the guide mechanism that travels along the lead screw is supported and does not twist around the screw. These guide rails are threaded into the end plate and extrude through into the enclosure where threaded nuts are attached to the threads. This is to ensure the guide rails are fixed sufficiently. Shown below is the schematic for these rails.

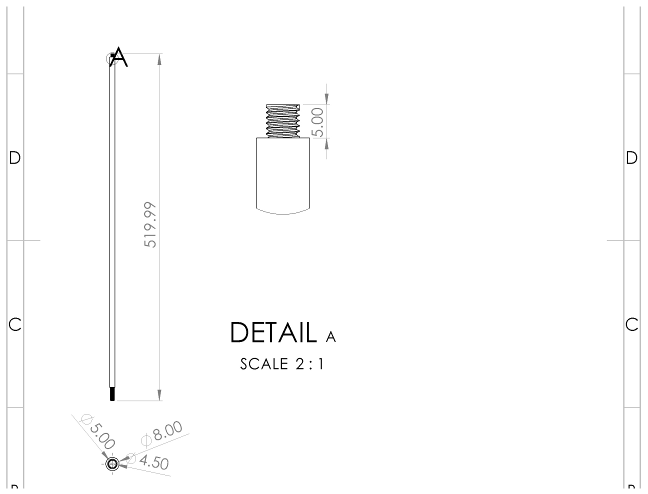


Figure 2: Guide Rail Technical Drawing

3.2.5 Guide

The guide mechanism is the mechanism that allows for the mounting of the mount plate. It is also the mechanism that travels along the lead screw. As the lead screw turns the guide mechanism is displaced 3mm for every rotation. The guide is also complete with 8mm diameter cut extrusions that for the guide rails in keeping the guide balanced. Shown below is the schematic for the guide

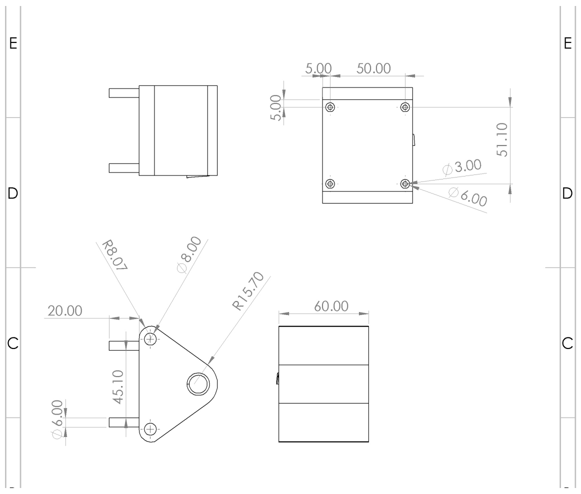


Figure 3: Guide Technical Drawing

3.2.6 Base

The base consisted of 4 X 4mm holes that allow for long bolts to be threaded into. This ensures the enclosure and the base with the enclosure cover are fixed together. The base also allows for the cover to rest upon it. The alternate end of the base consists of identical threaded holes where the end cap is mounted to the base.

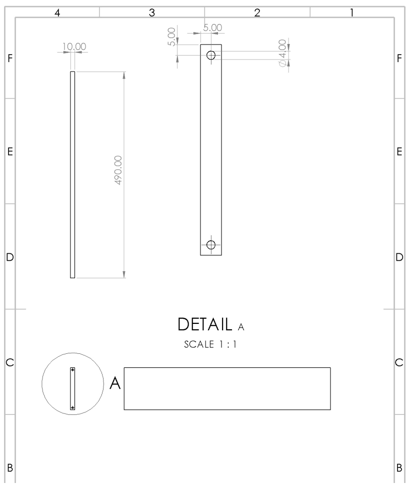


Figure 4: Base Technical Drawing

3.2.7 System cover

The system consists of internal threads on each side and allows for both the long bolts and the short bolts to be threaded into which completes the structural aspects of the system. These internal threads are bushes that are pressed into the material. In addition to this the cover contains slots that allows the guide to move along the threaded rod without restrictions. Shown below is the schematic for this component

3.2.8 PCB Bracket

The PCB bracket was designed so that the edges of the board slide into the bracket. The edges of the bracket ensure that the pcb is held securely. The bracket is then mounted to the enclosure cover with 4x nuts and bolts.

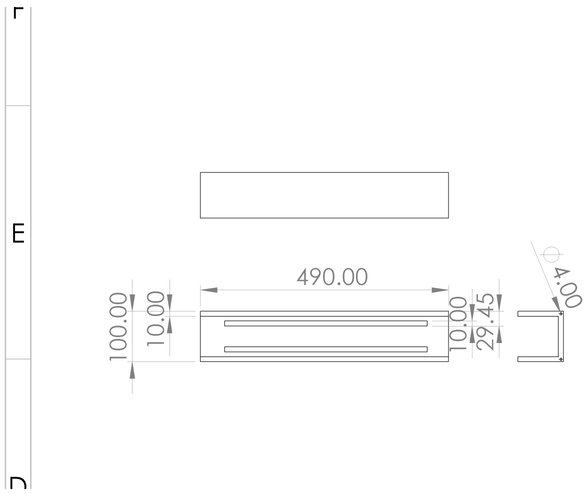


Figure 5: Cover Technical Drawing

3.2.9 Enclosure lid

The lid for the enclosure is to be screwed onto the enclosure once the stepper and pcb have been installed. This lid also consists of mounting holes for the pcb bracket which is shown below in the schematic. The lid is the last component to be installed on the system and completes the final assembly.

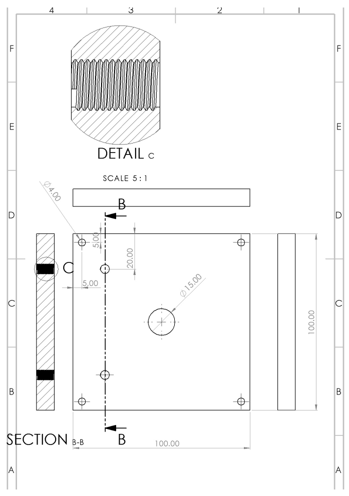


Figure 6: End Cap Technical Drawing

3.2.10 Mount Plate

Finally, the since the mechanical system is complete the last component to be mounted is the plate. This mounts directly onto the guide mechanism with screws. The screws are recessed to ensure there are no extrusion on the plates surface. Shown below is the final assembly for the mechanical system

3.2.11 Overall System

3.3 Results

The overall feasibility of the systems motion was carried out via a solid works motion simulation. This simulation consisted of rotating the stepper shaft so that the lead screw rotates which causes the guide to move along the lead screw. This causes a one-dimensional motion in both forward and reverse. However, this system is limited to only one dimension. I would suggest for future work making the one-dimensional system modular so that 2 degrees or even 3 degrees of freedom can be achieved.

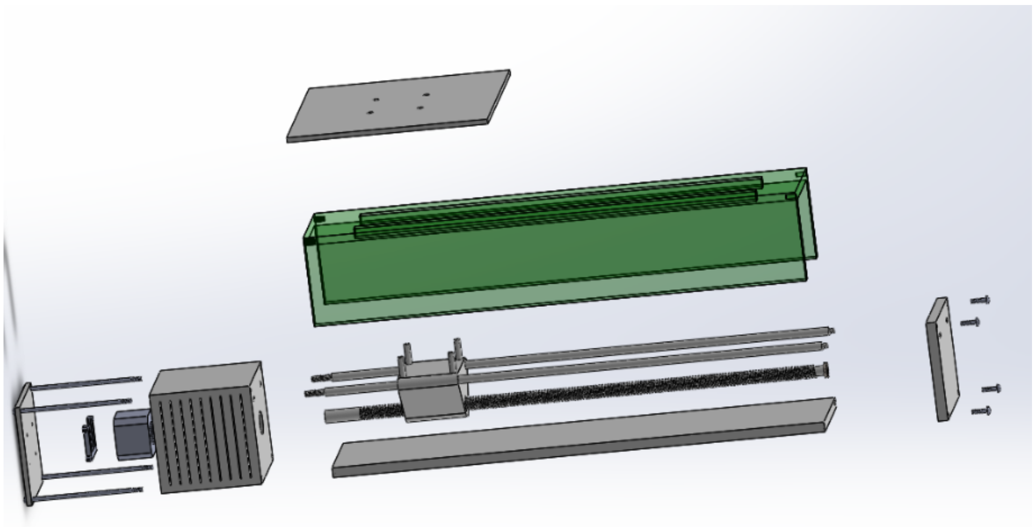


Figure 7: Exploded Version

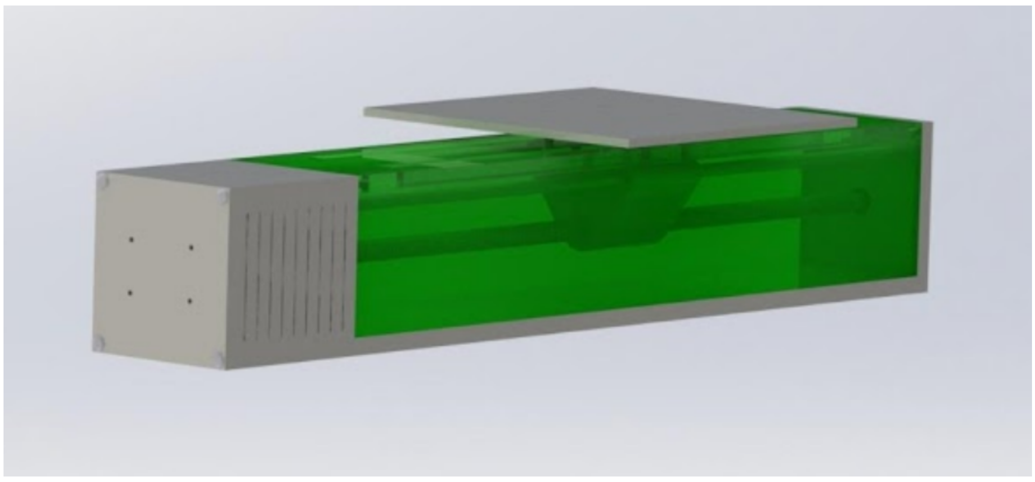


Figure 8: Side View of Assembly

4 PCB Design

4.1 Design Process

The PCB was designed to follow the specifications described above, as well as follow some specific constraints. First, it must fit within the size profile of the rest of the mechanical assembly.
Second, the PCB must have many power options, with flexibility for lots of different power input types. However, as separate voltages are required to run the motor and the microchips on the board, regulation should also be in place to provide internal power when external power is not provided.
Finally, to follow the constraints set above, the microcontroller was chosen to be an AT-Mega 32u4 microcontroller for its extreme ease of interfacing with USB, and a TMC2130 driver, for its incredible StealthChop technology, which allows steppers to run silently.

4.2 Methodology

The PCB was designed in KiCad, an open source PCB design software. KiCad was chosen for its ease in adding external libraries, as well as a wide availability of said external libraries. It is also heavily used by 'maker' communities, meaning that all of the parts that were chosen for this design were already available in the default install of the program.

4.2.1 Schematic

In the top left corner of Figure 9, the input terminals for motor power can be seen. It was chosen to use a screw terminal, a 12VDC barrel jack, and an XT-60 connector. However, the XT-60 connector does not have a

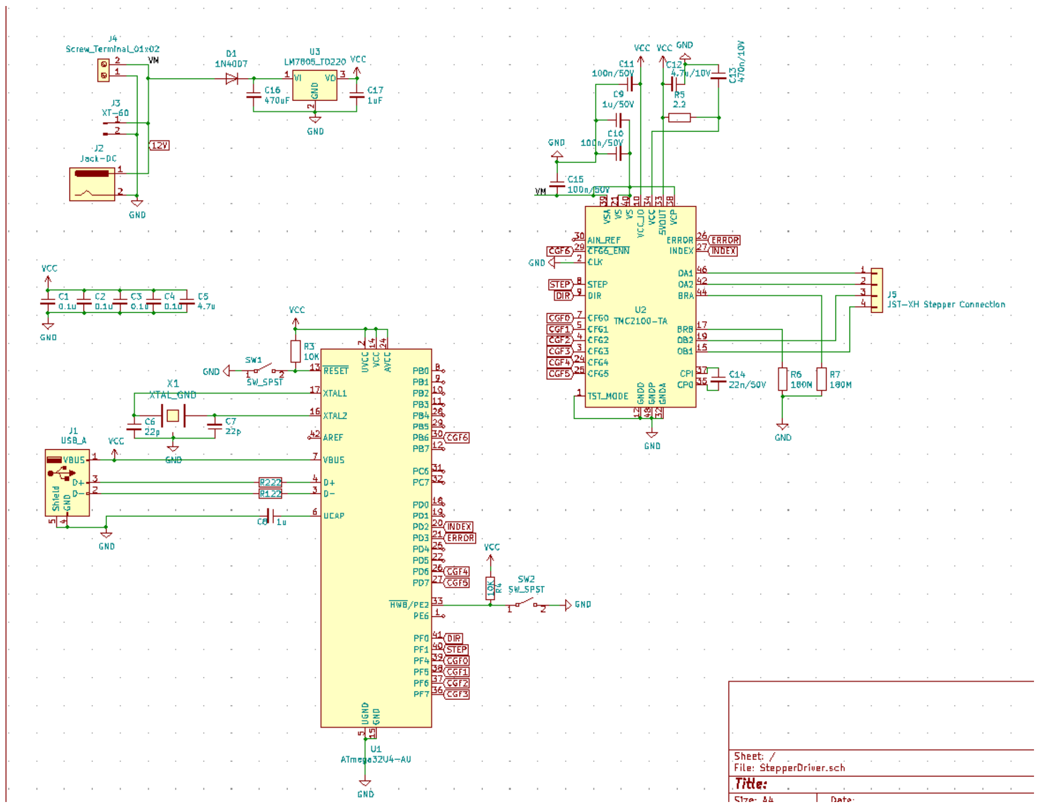


Figure 9: PCB Schematic

90-degree mount, so it is simply solder pads on the PCB. This means that if the final user wanted to, a different power input type could be used.

To the right of the power inputs, a LM7805 voltage regulator can be seen, which regulates 7-35VDC to 5VDC. The jacks discussed above are 12V standard, however, higher voltages could be applied, meaing the full range of the regulator could be used.

Below the regulator and power inputs is the AT-Mega 32u4 microcontroller. Various critical circuitry is seen to the left, with reset, clock, and data inputs all seen on the left hand side. The top and bottom are power inputs for the device, with VCC coming from both the USB input and the regulator. To the right of the device is its I/O. These are entirely devoted to the driver, with exception of pin 33, or HWB. This is a special pin that when LOW when Reset is LOW, the microcontroller vectors to the boot loader, for flashing.

The TMC2130 stepper driver can be seen on the right side of the schematic. Significant smoothing and input power protection are necessary for safe device operation, and can be seen on the top of the microcontroller. The left of the device are the data and setup lines, and the right are the motor outputs, as well as current sense, and error indication pins. Output to the stepper motor is provided through a 4-pin JST XH connector, seen on the far right of the schematic.

4.2.2 Wiring Diagram

As can be seen in Figure 10, space is a significant constraint for this design. The PCB is a two layer design, with the I/O ports, as well as the stepper driver, on the top layer. The bottom layer is much simpler, with the microcontroller and its clock. Ground planes are used on both the top and bottom layers, to reduce complexity, and decrease the electromagnetic interference the PCB could generate.

Traces of 0.250mm were used for data, and 0.4mm traces were used for power. All via's are .8mm max diameter.

4.3 Results

KiCad can produce 3-D live renders of a PCB, which can be seen in Figure 11. The power inputs, reset and boot switches, and the stepper driver can be seen in the top layer, giving an indication of the scale of some

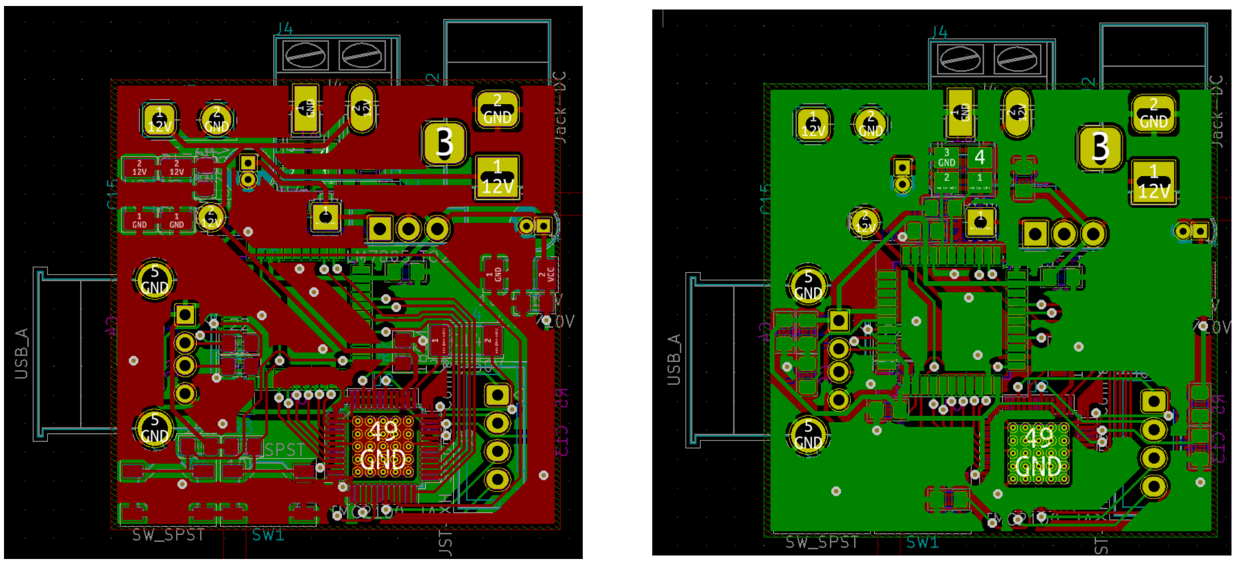


Figure 10: PCB Tracing, Front and Back Layers

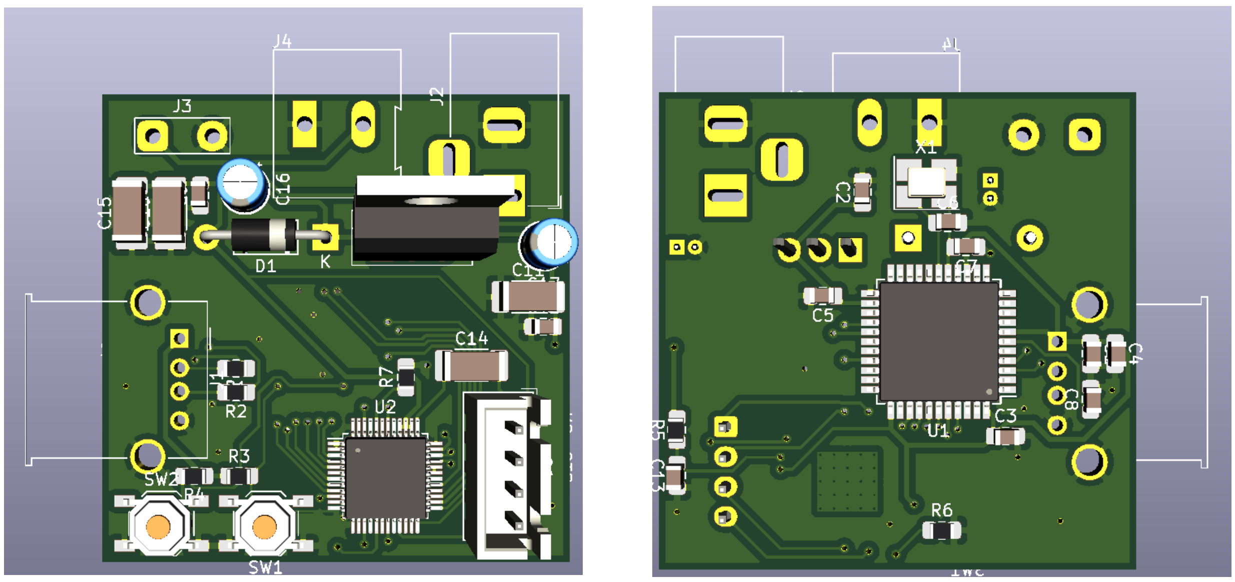


Figure 11: PCB Renders, Front and Back

of the devices used. The bottom layer primarily hosts the AT-Mega 32u4 microcontroller. An interesting note can be seen to the South-West of the microcontroller of the bottom layer. This square is the underside of the stepper driver. In order to reduce heat, it uses many via's to connect to the other layers, meaning a much larger copper contact area, resulting in increased heat dissipation.

Without testing the designed PCB, the primary concern is the positioning of the crystal clock. It is on the bottom layer, directly underneath the 12VDC barrel jack. This means that a sensitive piece of timing circuitry is underneath a high voltage (Relative to the rest of the board) area, which could easily experience voltage spikes when the supply is plugged in or out. This could result in clock fluctuations or compent damage.

5 Microcontroller Program

5.1 Design Process

The AT-Mega 32u4 chip is used as the microcontroller for the Arduino Leonardo, so Arduino's IDE can be used, with the board target set as the Leonardo. The pins were determined according to the Arduino reference, which also included the port numbers. This allowed for pin correlation from the schematic seen in Figure 9.
The program uses a 3rd party library, 'TMC2130Stepper', available on GitHub, or through Arduino's built in library manager. The library is extremely convenient, as the TMC2130 driver uses a custom control interface

based on timing high pulses for how long the stepper runs. Using the external library allows this process to be abstracted away.
The program also uses 'AccelStepper', again available on GitHub, or through Arduino's built in library manager. This library is used more for high level control over different steppers, allowing functionality like max velocity and acceleration to be set in code.

The TMC2130 has 6 configuration pins, which allow for hard wired configuration of various functions, like microsteps (if not set by the program), as well as hysteresis and clock values for various functions. These are hard set to HIGH or LOW at the start of the program, but could be changed for different driver functionality.

For communication with the controller, a basic communication protocol was established. The commands seen in Table 1 and 2 are relative to the controlling PC.

Sent String	Control Process
'O'	Enable Toggle
'D'	Direction Toggle
'F X'	Fullstep for X steps
'M X'	Set microsteps to X
'X X.Y'	Go to estimated displacement X.Y (Absolute)
'Z X.Y'	Go to estimated displacement X.Y (Relative to current position)
'H'	Find Home
'S M X'	Set max speed to X mm/s
'S A X'	Set max acceleration to X mm/s
'S S X'	Set stall value to X

Table 1: Command String and Control Process

Received String	Control Process
'C X.Y'	Current estimated displacement / position of head X.Y
'E'	Error detected
'N'	Home position / Current overload

Table 2: Received String and Control Process

5.2 High Level Code Flow

The program begins by defining pins, then performing configuration setup for the stepper driver and control libraries. It also opens a serial and SPI port, for communication with the controlling PC and the stepper driver, respectively.
The start of the program also sets up a clock based interrupt. The time between interrupts is used for control of how fast the stepper moves, however, this is mostly handled by 3-D party libraries.

The program then moves to the primary loop, checking the status of the driver and if there are any commands from the controlling PC.
Each of the command strings coming into the microcontroller, seen in table 1, are handled by a switch/case statement, and the serial port is checked for a new command every 0.5s.
The TMC2130 driver has multiple options for getting feedback on its current operational status. First, a 32 bit register can be requested over SPI, for an overview of various flags, like current draw, temperature status, short, overload, and step status. In addition, there are also two direct output pins, ERROR and INDEX. INDEX is pulsed HIGH when the stepper connected completes a full revolution, and ERROR is set HIGH if any error conditions occur, such as undervolt, overtemp, short, or step loss.
The status register is checked by the microcontroller every 0.1s, to check for overcurrent. This allows a normal operational current limit to be set, then, the driver to stop if the current limit is exceeded. Practically, this means that this can be used for endstop / limit detection, when the current values become too high. This allows calibration of the stepper without limit switches. When overcurrent is detected, the program assumes that the stepper has 'homed' and sends this signal back to the controlling PC.
The ERROR pin is monitored with an interrupt pin, and simply tells the stepper to shutdown and communicates back to the controlling PC.

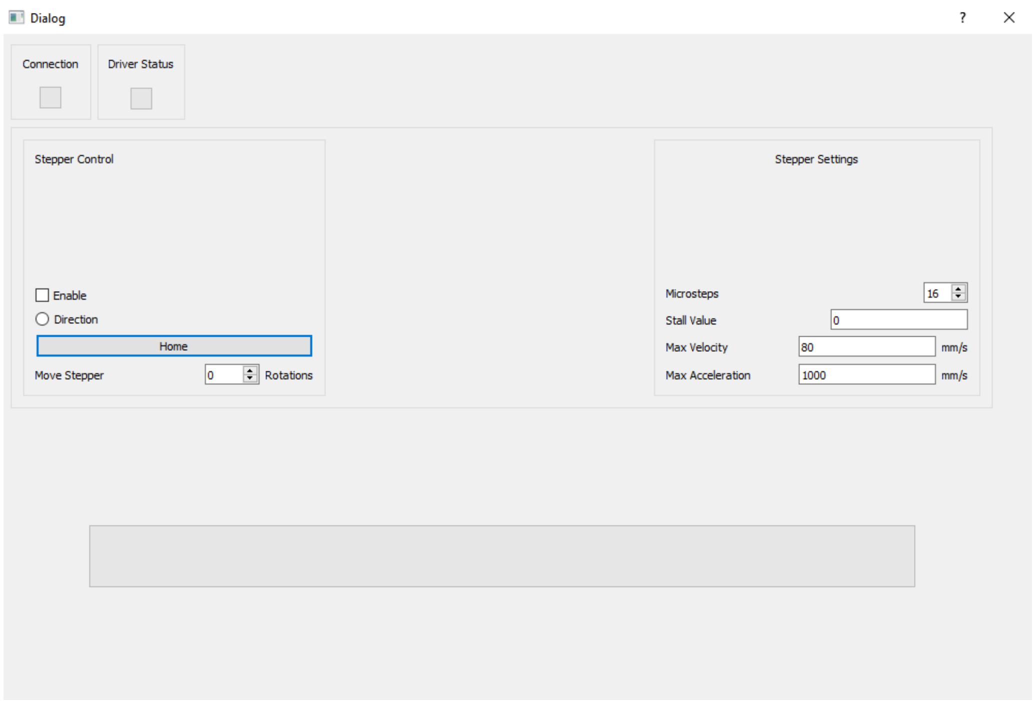
5.3 Results

Without testing, it is extremely difficult to know if this program would work as intended. The program compiles in the Arduino IDE, however, testing is extremely difficult. There are also 2 examples of blocking code, which occur when the stepper is asked to move to a position or home. This is bad programming practise, however, more complex solutions would be much more difficult to implement and debug, especially with several Interrupt Service Routines already being used.
There are a couple of features which were unable to be implemented on this 'theoritical' build of the program. The features missing are the ones which involve moving the stepper to an X.Y position. Due to the control nature of the TMC2130, which uses time as the controlling factor, as opposed to a LM298N, with which the microcontroller can directly count the steps, it is much more difficult to estimate the work head's displacement. With a prototype model, the movement time could be measured, and a formula for estimating the displacement could be used, however, without a physical model, this is impossible.
The program would likely need more development and debugging before final deployment.

6 PC Control Program

6.1 User Interface

The control UI can be seen in Figure 12. It is designed and implemented in QT, and implements all the features except for positioning at a X.Y displacement. The base of this program comes from the base provided for assignment 6, by Dr Peter Kay, in the second year Hardware Oriented Programming course.



Microsteps 16
Stall Value 0
Max Velocity 80 mm/s
Max Acceleration 1000 mm/s
Move Stepper 0 Rotations

Figure 12: PC Control UI

From top left of Figure 12, the connection indicates the status of the Serial connection to the microcontroller. It is green when the serial port is open. Next to it, the driver status light is the inverse of the ERROR pin on the TMC2130. When an error is communicated up the serial port, this light turns out, and the stepper is disabled. It also turns on when the serial port is established.

In the next row, there is the stepper control section. This section has a toggle button for the enable state and the direction of rotation of the stepper. Below that, there is a button to re-home the stepper, which will move the stepper to the home position, or, the closest the work head can get to the stepper, limited by the mechanical assembly. Finally for the section, there is a section to move the stepper by an amount of rotations. This amount is relative to the stepper's current location, and can be entered as both a positive and negative value.

To the right is the stepper settings window. In this, numerical values can be entered for the max velocity and acceleration, as well as stall and microstep values. These default to the values programmed on setup in the microcontroller.

On the bottom is a progress bar, meant to indicate the current position of the stepper head. This feature is not implemented, as the microcontroller cannot currently estimate its displacement, therefore does not feed this information back to the controlling PC.

6.2 High level PC control

Using the slots functionality of QT makes the back end for the PC control software extremely basic. Most commands simply write to the serial port and return.
The most complex part is checking if an inputted value is negative. The system handles this by sending the command to reverse the direction of the motor, move the specified amount, then toggle direction again, leaving the stepper in the same state as when the program started.
As enable state checking is done by the stepper driver, there is no need to check for enabled state before sending any control commands.

6.3 Results

The PC control system is extremely basic, and most of the heavy lifting is done by the stepper driver. Testing and development with the full mechanical solution is necessary before program deployment, however, the basic functionality is present. There are additional features, such as COM port selection and display, settings import from controller and file, as well as dummy checking that could be added for minor increases in functionality.

7 Conclusion

This report has summarised the work performed to design and program a system for a 1-D actuator. Without physically building the parts described above, it is difficult to verify the full systems functionality. To help offset this, good design processes have been followed and explained above. In addition, high level specifications helped drive the design direction of this solution. More testing of various aspects are likely required before a final deployment could be made, and additional features could be added to help with user interactivity and ease of use. This assignment has helped us develop our mechanical, electrical, and software design skills, and, through the use of modern consumer electronics, we have developed a solution which we believe would work effectively and fulfil the design specifications.

In reflection, the choice of driver for this solution did not lend itself towards direct manual control from a PC. The drivers selected are designed to be used in 3-D printers, and other plotter type designs. As such, they work on different controlling principles, and are not intended for direct individual control. This made it difficult to implement some features without a physical test platform. However, the additional features of this driver still make it an awesome choice for the physical performance of the solution, once the software is strong enough to support successful driver operation.

8 Datasheets

The TMC2130 driver datasheet can be found here:
https://www.trinamic.com/fileadmin/assets/Products/ICs_Documents/TMC2130_datasheet.pdf

The AT-Mega 32u4 datasheet can be found here:
http://ww1.microchip.com/downloads/en/devicedoc/atmel-7766-8-bit-avr-atmega16u4-32u4_datasheet.pdf

The LM7805 datasheet can be found here:
https://www.sparkfun.com/datasheets/Components/LM7805.pdf