SHORT RUN BOX MAKER

A Senior Project submitted to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree of

Bachelor of Science in General Engineering

by

Jesus Valdez-Ruelas

June 2020

Abstract

Small businesses with short run product catalogs have trouble finding the right size box that will work for multiple products. In turn, this leads most businesses to use oversized boxes resulting in unnecessary shipping fees. The objective of this project was to develop and test a cost-effective prototype of a box making machine that could create a short run of custom sized boxes. A Co2 laser was found to be the best option for cutting cardboard and a vertically standing machine was designed to feed the cardboard via rollers. To control cardboard movement stepper motors were used and controlled using an Arduino Uno. A User Interface (UI) was developed using Excel VBA to communicate with the Arduino Uno and to pass on box size and type. The prototype proved to be effective in cutting cardboard patterns. Testing revealed the prototype could be twice as fast as manual cutting methods if an 80W laser tube or larger were used. The source code used to build this project serves as a good reference for future needs of accurate stepper motor movement and PC UI development.

Keywords: Cardboard, Laser, Cutting, Box, Making, Arduino, Uno, Excel, VBA, UI

ACKNOWLEDGMENTS

Thank you, Dr. Jean Lee, and Dr. John R Ridgely for your help.

LIST OF TABLES

LIST OF FIGURES

REPORT TABLE OF CONTENT

Introduction:

As the owner of a small short run furniture manufacturing business by the name of Central Coast Creations, I'm constantly faced with the issue of finding the right size box for the products I sell. As a result, many custom size boxes are hand made from large 4' x 8' sheets using steel blades, the process is both time consuming and costly. Based on 15 sample boxes made it took an average time of 8.2 min. It's estimated by sales that by the end of 2019 a total of 107 hours and $2,130 (at a rate of $20/hr.) would have been spent making boxes since the business started in 2017. However, scaling this up to a larger business dealing with a lot of custom products the cost of making custom sized boxes can be much more significant. In this senior project report, I document my research and testing of a short run box machine prototype as if it was to be implemented in a business like mine.

Problem Description:

Currently the method used to make custom sized boxes involves cutting a 4’ x 8’ sheet of cardboard using a steel blade. A pattern as shown in Figure 1 is cut out to the dimensions of the box that is to be made. Next, the flaps are folded on an edge of a table and the box is taped together to its final shape.

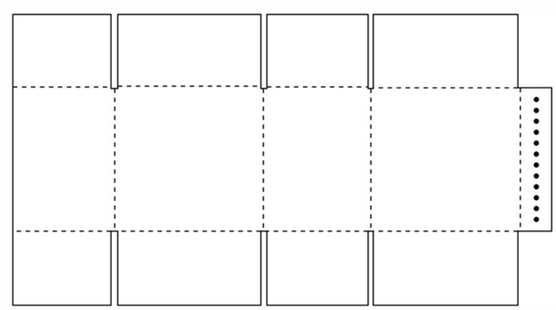


Figure 1: The standard box pattern (Wybenga, 2013, p. 470)

Due to the unpredictable shape of the products, most of them being handmade rustic pieces, a standard box size cannot be implemented for a single product. In addition, shipping costs for oversized boxes can become expensive if accumulated over a year.

Based on 15 sample boxes that were manually made it took an average time of 8.2 min to create a box. Using sales data from 2018 as shown in Figure 2 a total of 322 custom packages were made, this equates to a total of 44 hours making boxes. At a rate of $20/hr.

the cost of making boxes in 2018 was $880. While not very significant, by the end of 2019 an estimated total of 107 hours and $2,130 would have been spent making boxes. By accumulating this cost over the span of 5 years with business growth, the problem can reach a cost of over $10,000. Figure 3 shows a fishbone diagram of the causes of the problem.

Figure 2: Estimated number of packages to be shipped in 2019

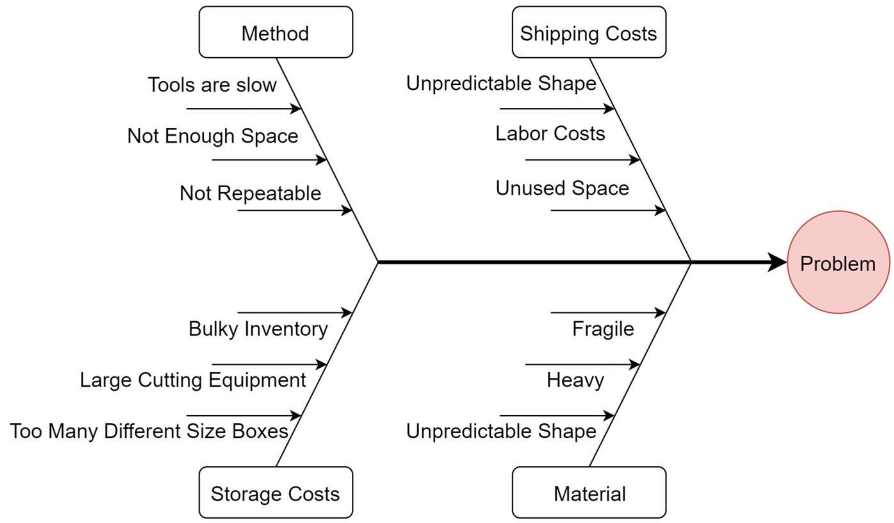


Figure 3: Fishbone diagram of the problem.

Literature Review:

To design a solution capable of creating custom sized boxes it was necessary to review both the cardboard manufacturing processes and test methods. From this knowledge the best design decisions were made later in the project.

According to Wybenga and Roth (2013) cardboard is made from two paper faces glued to a corrugated medium center using large glue rollers, this can be seen in Figure 4. While the figure shows how a single wall corrugated sheet is made similarly double and triple walled corrugated sheets are made, see Figure 5 for an example of cardboard structures.

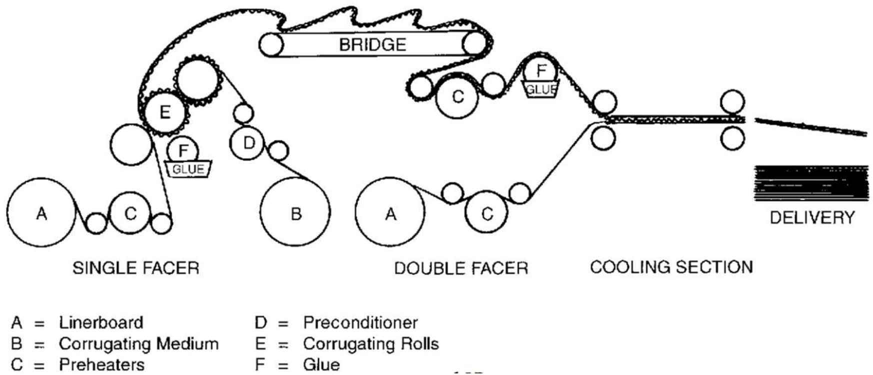


Figure 4: Cardboard manufacturing process (Wybenga, 2013, p. 462)

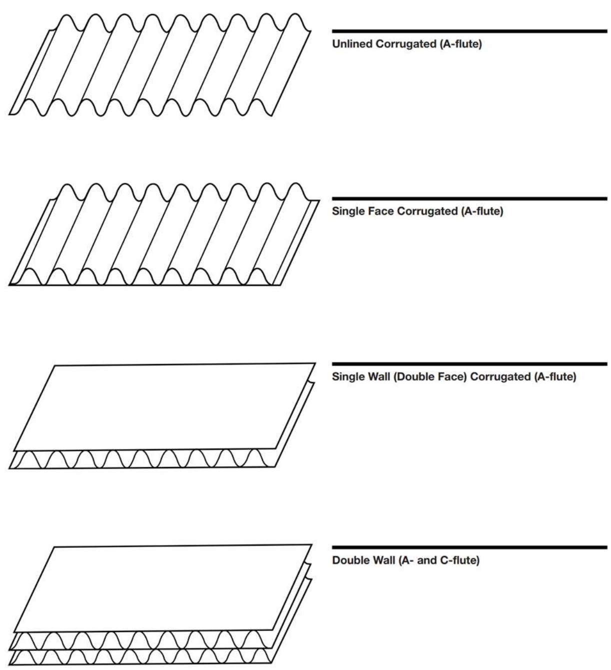


Figure 5: Most common cardboard structures (Wybenga, 2013, p. 463)

Internally the corrugated medium or flutes as it's called can vary in pitch and height depending on the desired structure. The flutes are usually indicated by a letter see Figure 6 for common flute types. On the outside of the cardboard sheet the paper that is bonded can vary depending on its use however, the most commonly used paper is virgin Kraft paper. For other commonly used papers in the industry see Figure 7.

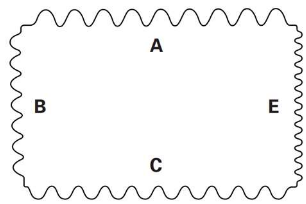


Figure 6: Sample flute types (Not to scale) (Wybenga, 2013, p. 462)

- KRAFT (K): Virgin Kraft paper
- TEST 2 (T2): Partly recycled liner paper
- TEST 3 (T): Fully recycled liner
- CHIP (C): Waste based liners
- FULLY BLEACHED WHITE (BW): Fully bleached Kraft liner
- WHITE TOP (WT): White coated recycled liner
- MOTTLED KRAFT (MK): Mottled white Kraft
- OYSTER (OY): Mottled test liner
- SEMI CHEM (SC): Virgin fibres using neutral sulphite semi-chemical process
- WASTE BASED (WB) 100% recycled fibres

Figure 7: Most common paper types (Daggar, n.d.)

Using cardboard sheets box manufacturers cut cardboard box patterns from them using large die cutters see Figure 8 for a sample box pattern. For more information on box patterns see Wybenga and Roth (2013).

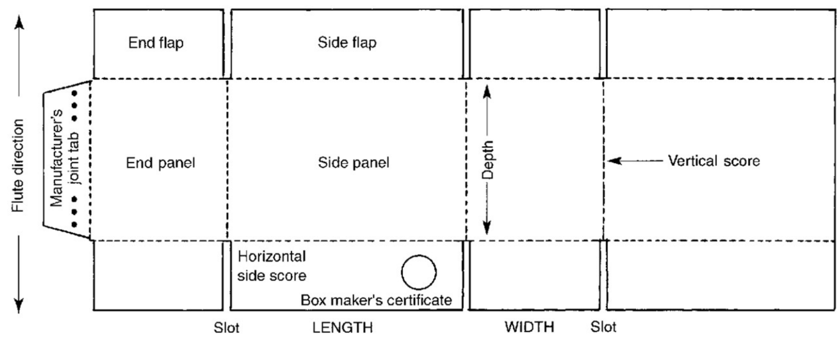


Figure 8: Sample box pattern (Wybenga, 2013, p. 467)

Usually box manufacturers attach their box certificates at the bottom of the box that usually contains information regarding weight limits, ECT (Edge Crush Test) results, and Burst test results. See Figure 9 for an example box certificate.

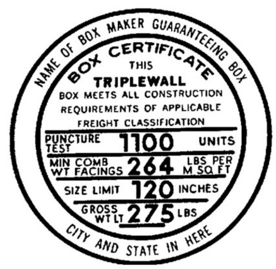


Figure 9: Box certificate example (Wybenga, 2013, p. 465)

Box manufacturers attach max weight limits before failure however, a safer threshold is desired to avoid shipping damages. Using UPS (United Parcel Service) strength guidelines the maximum shipping weight can be determined using the cardboard classification. See Figure 10 for UPS strength guidelines. Since this project is aimed towards small business owners using the most cost-effective solution and readily available solution only two types of corrugated sheets will be considered in this project, one being a single wall of ECT 40 and a burst strength of 200 psi as well as a double wall of ECT 48 and a burst strength of 275 psi. All other styles of cardboard are either too expensive to be effectively implemented or not readily available and must be special ordered.

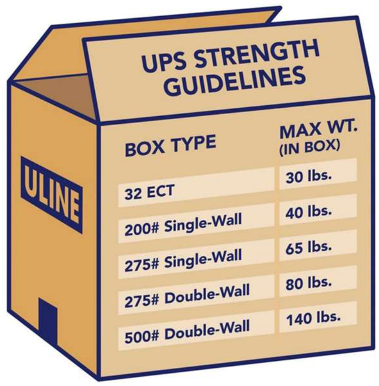


Figure 10: UPS strength guidelines **(**Uline**)**

While researching other custom box machines were found but none that were both cost effective and implemented automation at a small scale. With a growing number of custom product sellers in ecommerce there is an increasing need for a solution. If a solution is implemented buyers can pay less for shipping, and sellers can increase profit margins.

Solution Design:

While gathering ideas for how the design of this machine will look like two layouts were taken into consideration, one being in the form of a gantry machine and the other being in the form of a large format printer see Figure 11. Due to the layout of a gantry machine it can be less complex to add different tools to the machine. In addition, a gantry machine would not require a method to feed in the cardboard because most machines such as CNC (Computer Numerically Controlled) routers use vacuum tables. However, since space is very important because this project is geared towards small businesses a large format printer layout is desired. Both layouts would require space for at least one cardboard pallet and the machine. As shown in Figure 11 a large format printer layout takes significantly less space when compared to a gantry machine. At a normal market rate of $1 per square foot for a warehouse lease, the cost for a gantry machine would equate to $540 a year while a large format printer layout would equate to a quarter of that cost. As a result, a large format printer layout was decided early on. An early concept of the design can be found in Figure 14:. However, to further reduce space the final design of the machine resulted in a vertical layout such as to have it laid up against a wall. In practice this reduces the amount of space that needs to be in front of the machine and creates a smaller footprint when compared to a large format printer.

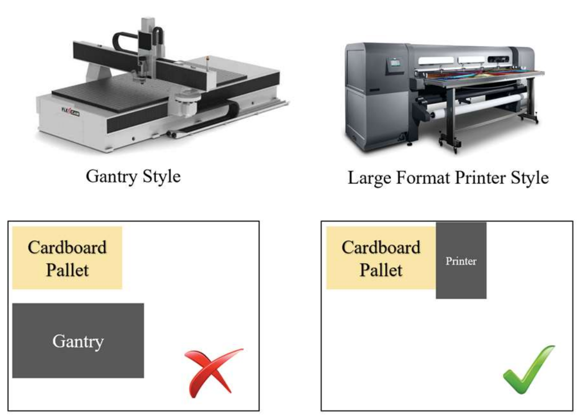


Figure 11: Space comparison of shop floor between a gantry layout and large format printer layout.

While ideating forms to cut the cardboard and researching cutting methods Mathilde (2014) was a good source. The following decision matrix was made to compare different methods. Higher numbers are the most desired aspects such as lowest cost, quickest speed, highest safety, longest life, and least complexity. All aspects are equally important and are weighted equally in this decision matrix.

Table 1: Cutting method decision matrix

Cutter	Cost	Speed	Safety	Life	Least Complexity	Total
Laser Cutter	2	10	2	10	10	34
Steel Blade	10	2	8	2	2	24
Rotary Cutter	9	5	8	4	4	32
Die Cutting Blade	1	10	1	9	6	27

Initially it was thought that a rotary cutter would be the best option for this project but proof of concept testing later revealed that a complex method of holding the cardboard againts the rotary cutter would be required. As a result, rotary cutters and steel blades were ruled out as a possible solution. Die cutting blades were considered but were also ruled out do to the cost of blades they would be out the budget of a low cost soultion. Additionally, the size of the machine would have to increase to move the blades in and out of the cutting area this would be an undesirable feature. A laser cutter was found to be the best choice for this project, because It proved to be the simplest method to cut cardboad while being a fast solution. The fold edges could be cut using patttern lines to ease folding while still maintaining the structure of the box. However, the downside to laser cutting methods would be the fume extraction equiptment needed for safe operation and the initial cost of equipment. Due to time constraints the focus of this project was only on the the design of the machine, a fume extraction method was not designed as there is many products that do this already in the market.

While idealizing ways to feed in the cardboard to the rotary cutter during the initial concpet design two possible solutions were compared, electric and pneumatic actuators. After reading a comparison article written by Robert Kral (2015) an engineer for BIMBA® a major manufacturer of both electric and pneumatic actuators the pros and cons of each was listed and compared see Figure 12.

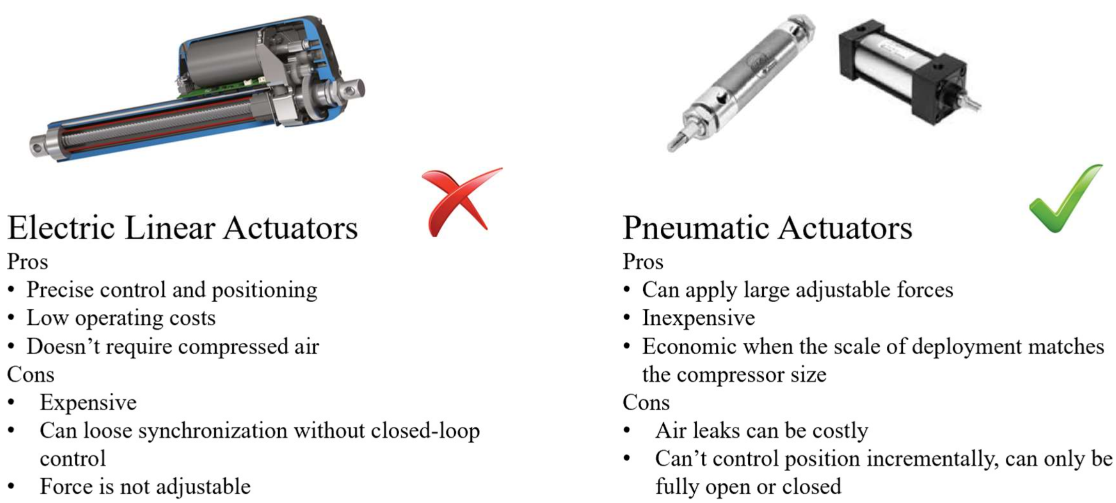
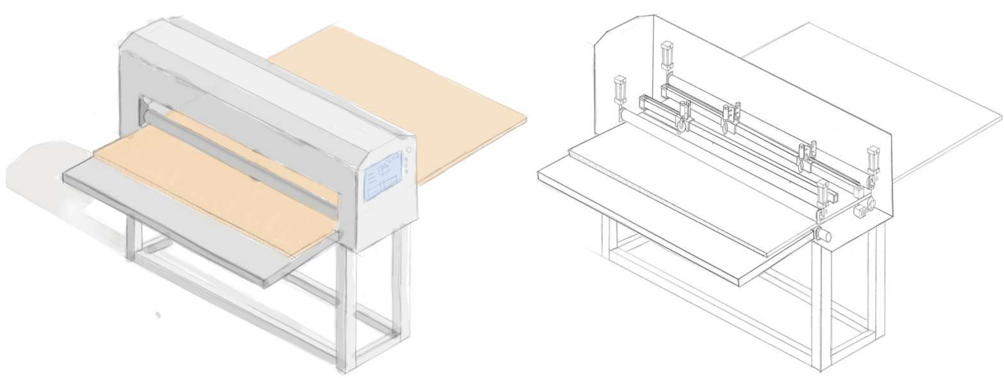


Figure 12: A comparison between electric and pneumatic actuators

Due to the need of regulating the force applied by both the rotary cutter and the cardboard feeder into the cardboard a pneumatic actuator was chosen for the early proof of concept prototype. Since this project is geared towards small business and shops it ws assumed that compressed air was readily available. However, do to the project switching its course over to a laser cutting method actuators were no longer required.

Furthermore, using large format printers as an inspiration large rollers were chosen to feed the cardboard into the machine an early conecpt design is shown in Figure 13. The anti-slip surface material was chosen to reduces the possibility of slippage and skipping of the rollers hence letting the machine cut more accuratly. Testing later revealed neoprene rubber would be the best material and it was added to the final design. To drive the rollers a set of spur gears were designed to replicate the number of steps per inch required for the other axis. As a result, a 13:6 gear reduction created approximately the same amount of steps per revolution.

Figure 13: Cardboard feeder design decision



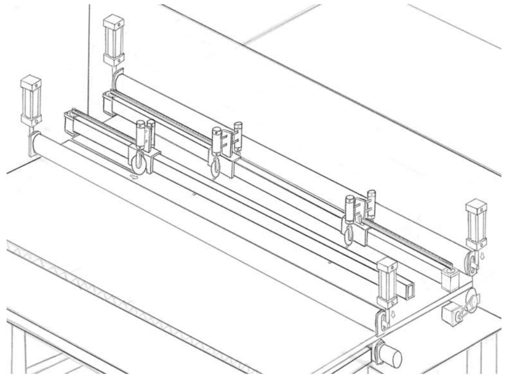


Figure 14: Early concept designs

When choosing a laser cutter, a Co2 laser tube was preferred over a laser diode due too their high wattage and fast cutting capability. They are commonly used to cut thin plywood sheets for arts and crafts but can be used to cut a wide range of materials including cardboard. As a result, a 40 W laser tube was chosen for this project. A more powerful laser tube would increase cutting speeds however for the purpose of creating a prototype a 40 W laser tube was determined sufficient. The mirrors, focal lens, and mirror mounts required for this project were selected using off-the-shelf components. Early on it was determined that the machine would be designed around a 1.5" focal length the frame was designed to work with such lens. In a complete solution a fume extractor such as those used by Co2 laser machines would have to implemented with a hood to contain the fumes.

To control the movement of the laser lenses and the cardboard feeder Nema 23 stepper motors were chosen for their high torque, precision step movement, and low cost. Combined with micro stepping drivers the motors are capable of micro stepping 16 steps within their normal 200 steps per revolution. Such feature is desired to accurately cut the cardboard patterns. In addition, its high torque can be used to drive the cardboard feeder rollers at reasonable speeds.

To control stepper motor functions an Arduino Uno was chosen as its clock frequency of 16 MHz was determined to be sufficient to perform the functions of this prototype which included accurate stepper motor movement. Another feature that was desired was its ability to communicate with a PC via a USB serial port. This feature would allow the creation of an easy to use PC user interface that could send over the required box dimensions to the microcontroller.

Before coding began, the microcontroller pins were assigned for specific functions, it was also determined that two limit switches would be required to home the machine and to find the starting position, similar to how a CNC machine homes on startup. To have more control of the machine a button interface was deemed necessary to be able to quickly stop and reset the machine if it ran out of cardboard. Additionally, a push out function was desired if the machine got stuck with cardboard it could be easily pushed out. Pins for these buttons were also determined and can be seen in Figure 15 with the rest of the assigned pins.

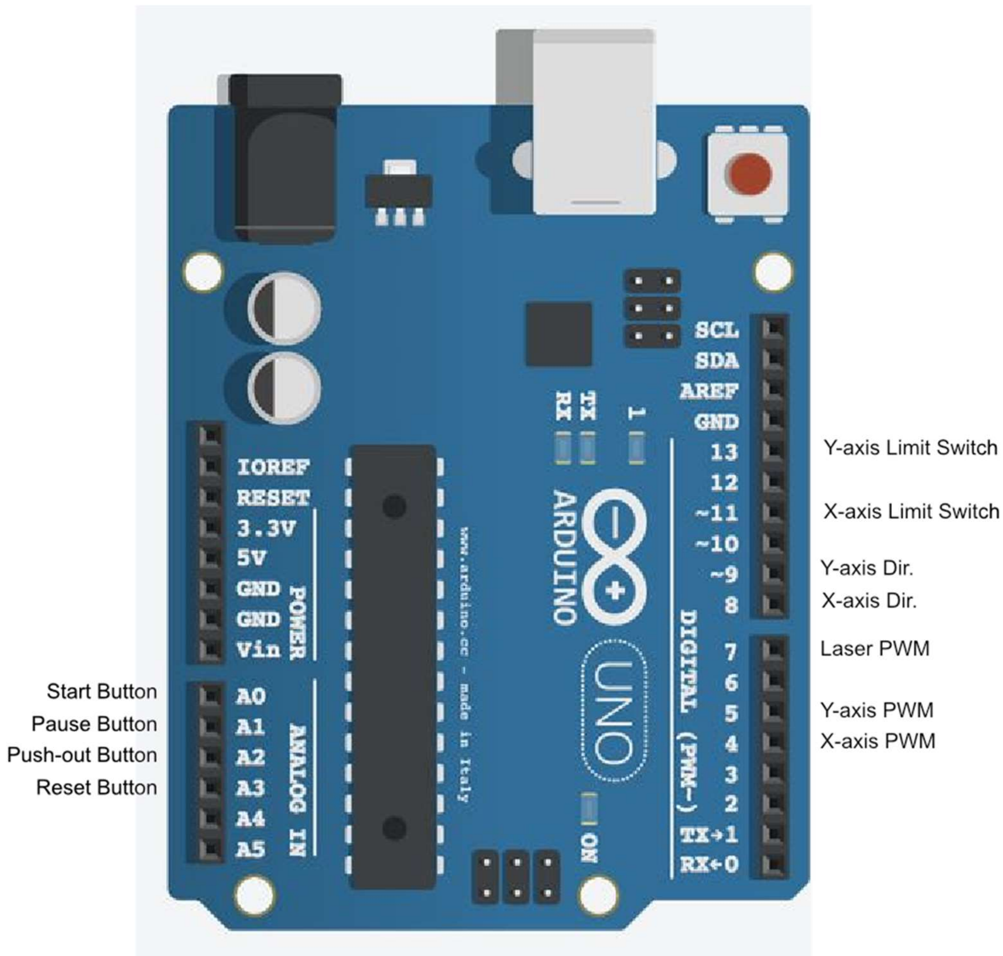


Figure 15: Assigned pins for the SRBM project

After assigning pins, state transition diagrams were made for the most important tasks they can be found in Figure 16. To reduce the number of instructions that would have to be passed on through serial communication most of the movement instructions were coded into the microcontroller. This reduced the number of instructions required through serial communication to 7. The format can be seen in Figure 17 commas separate the variables which then get turned into a list of coordinates for movement within the microcontroller.

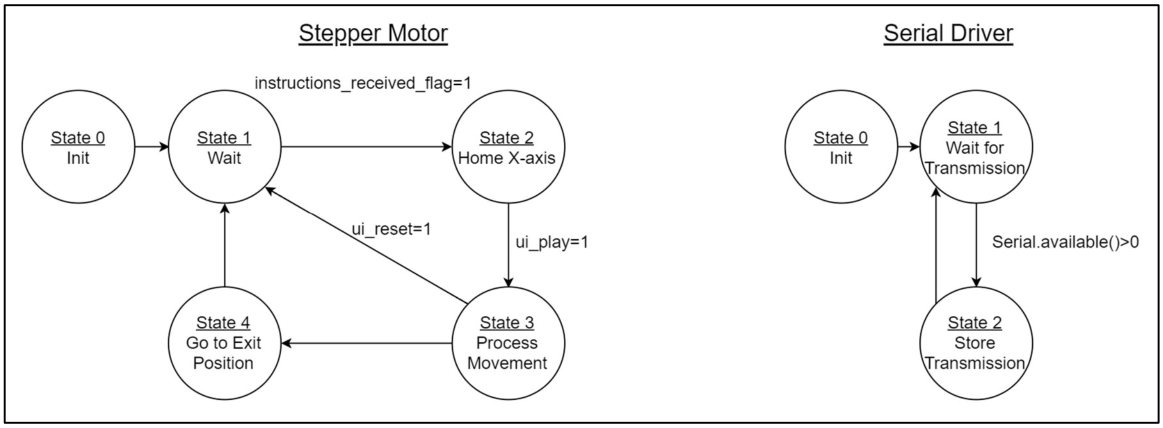


Figure 16: State Transition Diagram for the SRBM

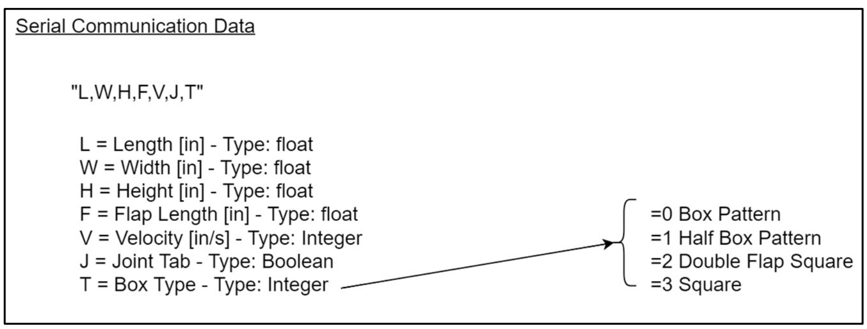


Figure 17: Serial Communication Format

Using Visual Basic for Applications (VBA) within Excel an easy to use interface was designed. The User Interface (UI) can be seen in Figure 18 a copy of the source code can be found Appendix 6. The UI was designed so the user could quickly select a box pattern and input the different dimensions. Four box options were designed into the system including the commonly known box pattern, half-box pattern, double-flap square, and a square. The half-box pattern was implemented for making boxes bigger then possible with a 4’x8’ cardboard sheet. The double-flap square and the square pattern were implemented to help with reinforcing heavier boxes or to separate internal box content. A joint tab option was implemented in the UI to allow easy removal of the joint tab incase the box would not fit into a 4’x8’ sheet.

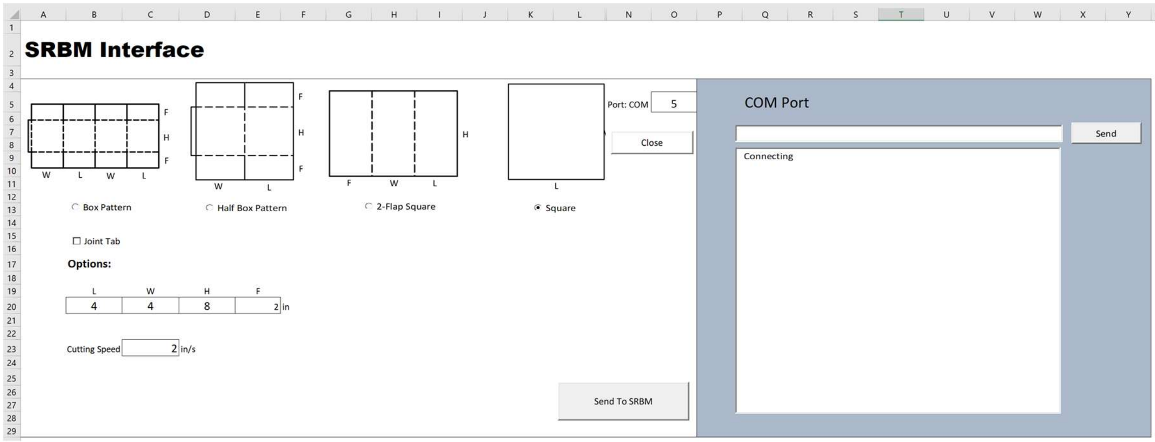


Figure 18: Excel User Interface

To smooth out stepper motor operation the pulse width modulation (PWM) required for stepper movement was controlled in an Interrupt Service Routine (ISR) which was programed to loop at 100 Khz the maximum frequency the stepper motor drivers could take according to the data sheet. Controlling the stepper in an ISR would make sure that the PWM was accurate, and without interruption from other controller functions. To be able to control stepper motor speed, calculations were made for the PWM required to reach a desired speed based on micro stepping, pulley teeth, and belt pitch configurations. These were then used in the ISR to output the PWM required for the desired speed. They can be found in Appendix 7. Due to the complexity of the ISR when a frequency was determined for its reoccurring calculations it was slowed down because of the numerous operations resulting in slower motor speeds then inputted into the system. Possible explanation of this occurrence could be improper setup of the microcontroller's timer counter disabling counting during an ISR. However, a quick solution was found by measuring the ISR actual frequency required to perform operations and then correcting for it in PWM calculations to obtain the desired speeds.

Furthermore, the stepper motors were found to vibrate a lot during testing as a quick solution to the problem a 10- step linearly increasing velocity profile was used for both accelerating and decelerating. Vibration was drastically reduced, and it allowed for smoother transitions between movements.

Test and Evaluation of Design Alternatives

Initially rotary cutters were considered for this project and two tests were performed to validate the proper function of them. One test similar to how a tomodynamometer is used to measure blade cutting resistance on fabric using ASTM standards (ASTM, 2015) measures the cutting resistance of the rotary cutter. It also provides different cutting pressures in the pneumatic cylinder to be able to find the required cutting pressure for both the single and double walled cardboards. See Figure **19** for a diagram of the experiment and Figure **20** for the experiment apparatus. Appendix 2 contains the collected experiment data. It was found that for double wall cardboard (ECT 48) the cutter drag force peaked at an average of 5.1 lbf and for single wall cardboard (ECT 40) the drag force peaked at 3.4 lbf. These results will be used later on to size the linear bearings for the carriages.

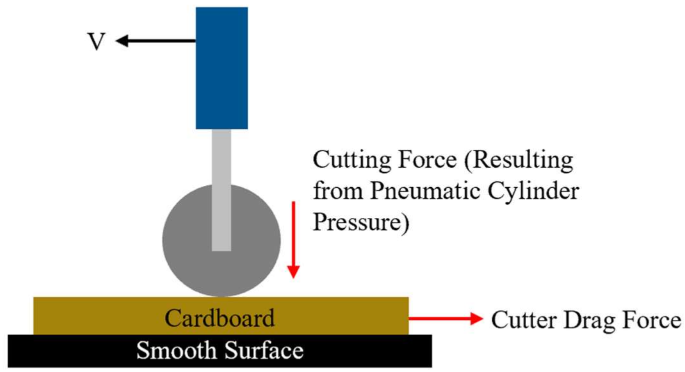


Figure 19: Diagram of cutter drag force experiment

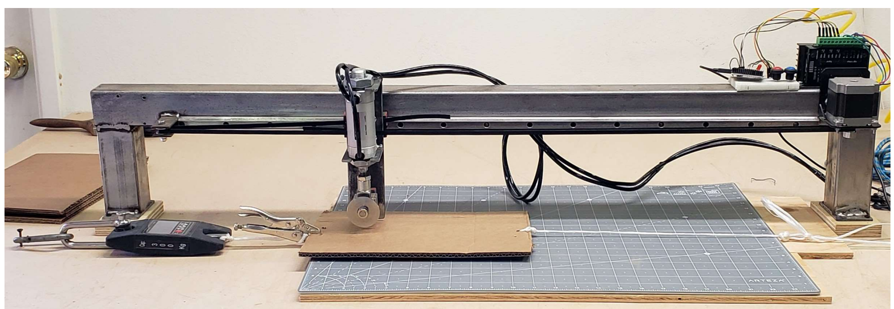
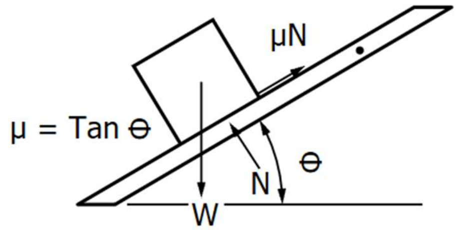


Figure 20: Experiment apparatus used to measure cutter drag force

The next test that was perfomed was used on the cardboard feeder anti-slip surface materials to find the coefficicent of friction. Using methods outlineds in ASTM standards (ASTM, 2018) different surface materials were tested for their coefficicents of friction on cardboard to find a sutible material for the feeder cylinder. See Figure **21** for the relationshp between maximum angle before slippage and coefficient of friction. See Figure **22** for experimental apparatus.



s

Figure 21: Relationship between maximum angle and coefficient of friction

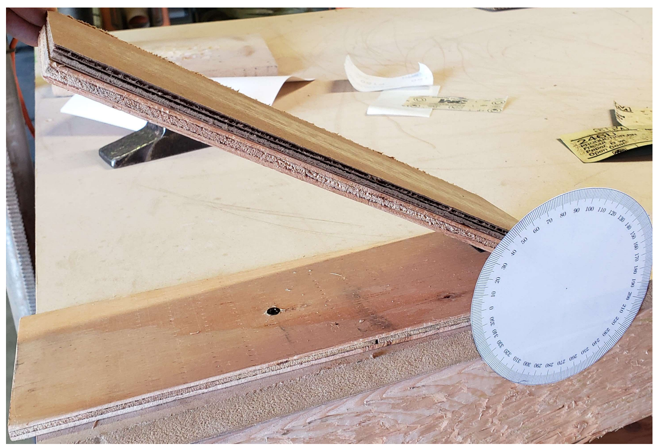


Figure 22: Experimental apparatus used to find the coefficients of friction

After experimenting with a few materials, neoprene rubber was found to have the highest coefficient of friction, see Table 2. Appendix 1 contains results for the coefficient of friction tests. Early on it was thought that sandpaper would have been a good option but due to its roughness and large scratches that were left behind on the cardboard test surface it was ruled out. From these results neoprene rubber resulted in having an excellent coefficient of friction while not damaging the cardboard surface.

Table 2: Average coefficients of friction between cardboard and sandpaper

Material	θ_{max} (°)	μ
Neoprene Rubber	46	1.0
36 Grit Sandpaper	48	1.1
80 Grit Sandpaper	46	1.0
150 Grit Sandpaper	44	0.97
220 Grit Sandpaper	42	0.89
360 Grit Sandpaper	36	0.73

To verify that the original concept design was going to effectively cut cardboard a simple carriage proof of concept prototype was built and tested as shown in Figure **23**. Next, a simple program was written in Python and was ran on a Pyboard microcontroller to activate the pneumatic cylinder using a solenoid as well as to move the stepper motor. The stepper motor driver can be found in Appendix 3. Through experimentation it was discovered that the rotary cutter blade was too thin to hold the cutting pressure and flexed to the point of curving the cutting line as shown in Figure **24**. Through experimentation it was found that by lowering the cutter into the cardboard and feeding it away from the lowest cutting edge the cutter could make a cut with less effort. To do this a slot was added along the cutting line as shown in Figure **25**.

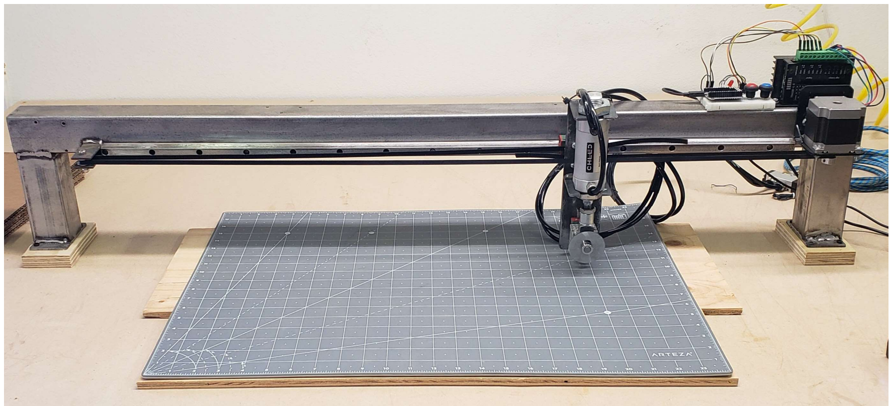


Figure 23: Proof of concept prototype used to verify proper function of design

Figure 24: Bending of cutter due to excessive cutting pressure

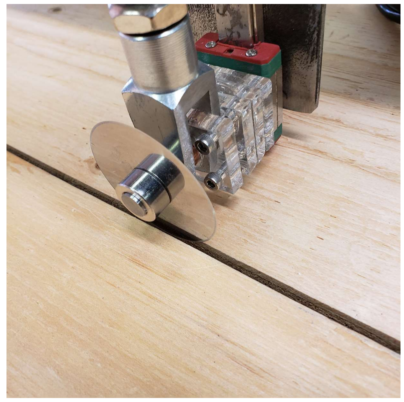


Figure 25: A slot that was added to raise the cardboard higher onto the blade.

After the proof of concept prototype was built using a rotary cutter it became clear that a solution using a rotary would need complex methods of holding the cardboard down against the cutting force. In turn, this led to the decision of using a laser cutter instead. Such design would not require pneumatic cylinders and would use significantly fewer moving parts.

Furthermore, the machine shown in Figure 26 was designed to feed in the cardboard vertically into the laser cutter. The wheels laying outside the machine were designed to feed in the cardboard sheet as straight as possible. They remain unattached to the vertical frame for easy moving and storage.

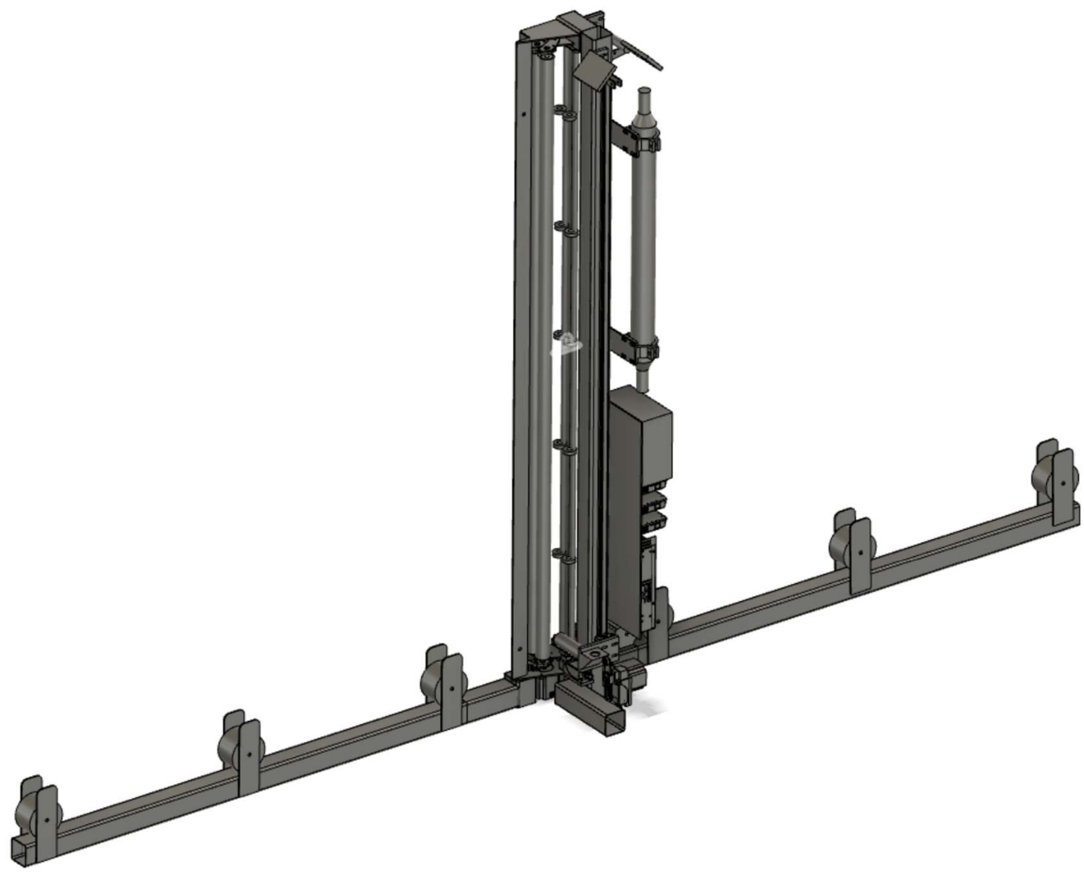


Figure 26: Final Machine Design

A prototype of this design was built and can be seen in Figure 27 a close up view can be seen of the front in Figure 28. The microcontroller was thoroughly tested for proper functionality. The machine was found to move the cardboard accurately and all button functions worked properly. However, due to improper concentricity of the roller ends to the centers the cardboard was found to slip slightly. This problem was due to a build defect and missing lathe equipment needed to make such part. The problem could be fixed with proper machining equipment.

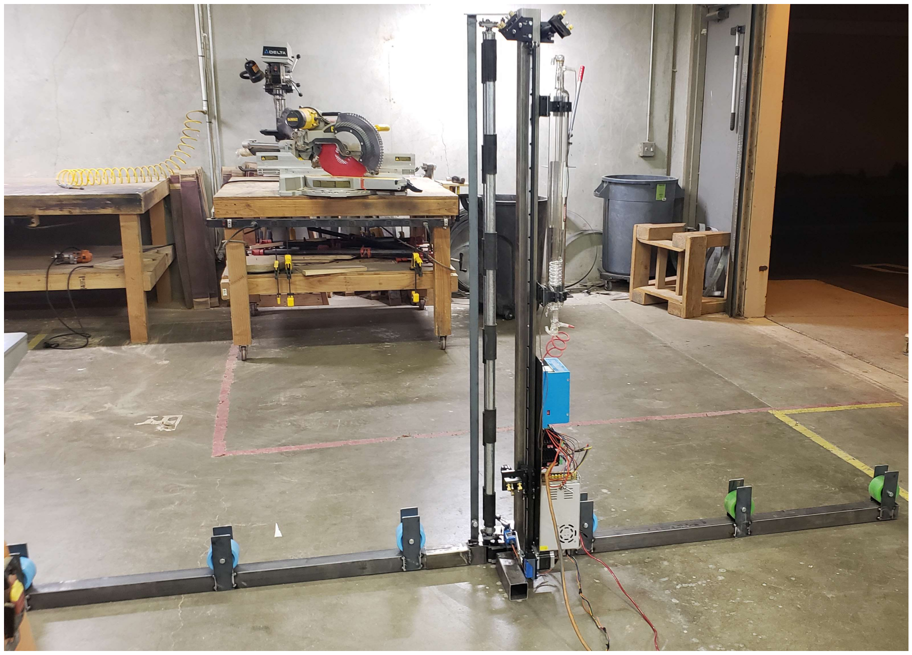


Figure 27: Final Design Prototype

Through testing it was determined that a cutting speed of .75 in/s was sufficient for double-wall cardboard and .1 in/s was sufficient for single wall cardboard. In theory at a cutting speed of .75 in/s a 24” x 24” x 24” box which is the largest box the machine can make would take 8.8 min. to be cut. When comparing this to the avarage time to make a box manually (8.2 min) it became obvious that a higher wattage laser tube would be necessary to operate efficiently, perhaps a laser tube with twice as much wattage such as an 80W laser tube. Then a 24” cube box could be made in less than 5 minutes. See Figure 29 for a close up view of cutting action.

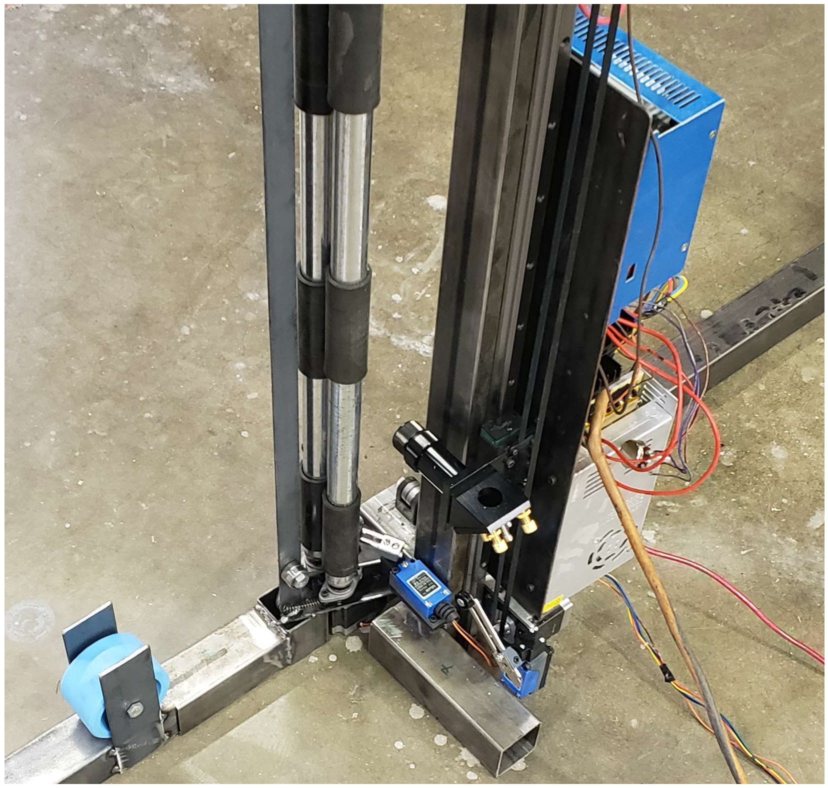


Figure 28: Close up view of final design prototype

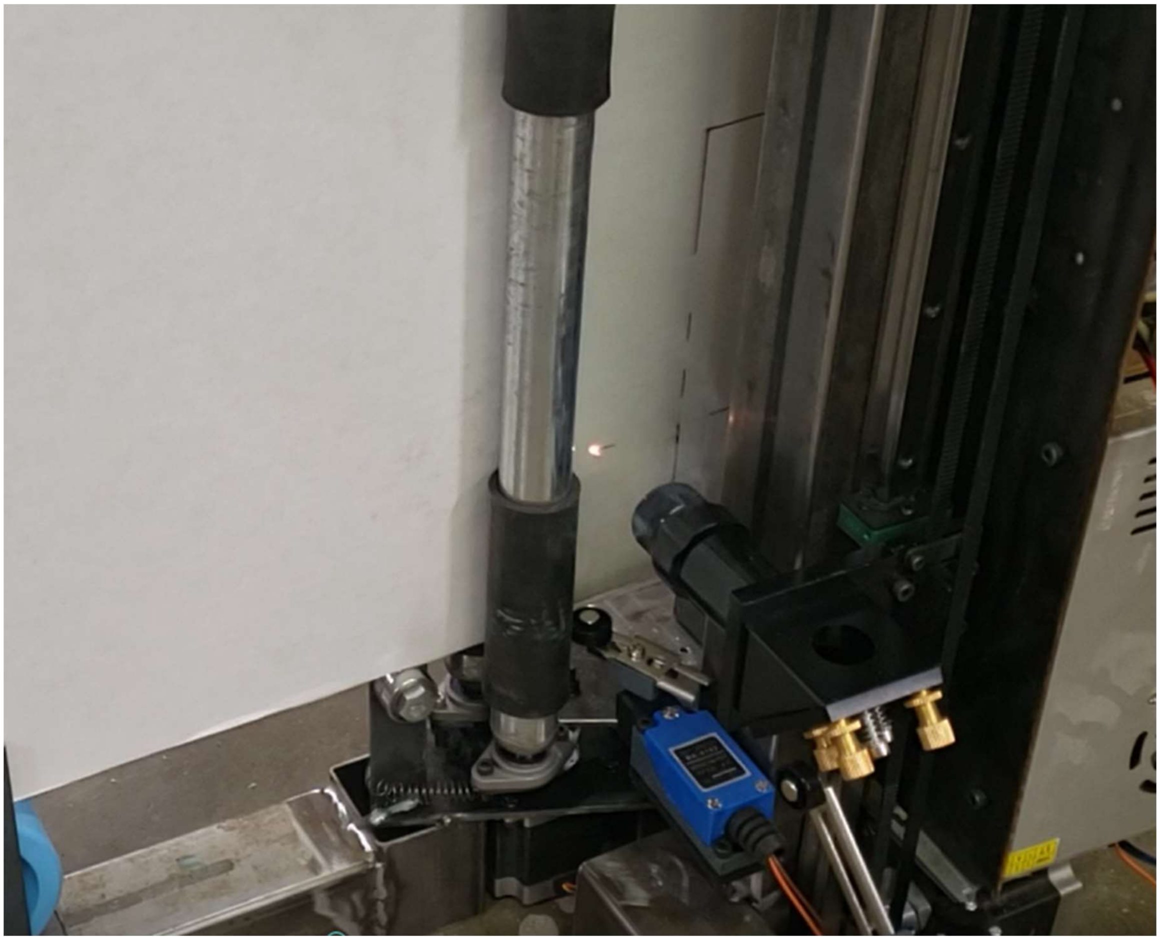


Figure 29: Close up view of cutting action

Conclusions and Recommendations

The final design of the short box maker prototype proved go be an effective solution to cutting cardboard in a small shop were a lot of custom sized boxes would be required. Its small footprint allows it to be placed up against a wall taking minimal space and the UI reduces the complexity of running the system. However, due to the number of parts required to build this machine it is still a complex machine to build. A gantry style machine would be recommended if the space permits due to it being the simplest design of all the options. Additionally, the UI and the microcontroller's programming can be replaced by using existing alternatives such as the open source Arduino g-code controller project called GRBL. Which can be fed g-code from existing pc software. A simple program written in G-code can be made with variables that can be edited quickly to change box dimensions. This reduces the build complexity and the programming skills required to build such machine significantly. However, the source code used to build this project serves as a good reference for future needs of accurate stepper motor movement and PC UI development.

Future Directions

If the need for making custom boxes grows to the point that a full-time employee would be needed to make boxes a fully autonomous machine could be designed. If space were not limited a machine could be built with a crane that could lift a cardboard sheet and could lay it on a gantry style machine. This machine could connect to an existing order database to cut boxes before they are needed hence reducing the need of human labor. However, such solution would require much research into API development.

References

ASTM. (2015). *ASTM F2992/F2992M-15 Standard Test Method for Measuring Cut Resistance of Materials Used in Protective Clothing with Tomodynamometer (TDM-100) Test Equipment.* Retrieved from ASTM International: https://doi-org.ezproxy.lib.calpoly.edu/10.1520/F2992_F2992M-15

ASTM. (2018). *ASTM G115-10(2018) Standard Guide for Measuring and Reporting Friction Coefficients.* Retrieved from ASTM International: https://doi-org.ezproxy.lib.calpoly.edu/10.1520/G0115-10R18

Daggar, J. (n.d.). *What Are Corrugated Board Grades?" GWP Group, www.gwp.co.uk/guides/corrugated-board-grades-explained/.* Retrieved from GWP Group: www.gwp.co.uk/guides/corrugated-board-grades-explained/

Kral, R. (2015, February 3). Electric vs. Pneumatic Actuators. *ASSEMBLY*.

Mathilde. (2014, November 19). *How to Cut Cardboard*. Retrieved from Making Society: http://makingsociety.com/2014/11/how-to-cut-cardboard-prototyping/

Twede, D. e. (2014). *Cartons, Crates and Corrugated Board: Handbook of Paper and Wood Packaging Technology.* DEStech Publications.

Uline. (n.d.). *box_weights.* Retrieved from Uline: www.uline.ca/images/en-US/CustomerService/box_weights.gif

Wybenga, G. L. (2013). *The Packaging Designer's Book of Patterns.* Wiley.

Appendix:

1. Coefficient of Friction Test Results

Table 3: Coefficient of Friction of Cardboard and Sandpaper Test Results

Material	θ_{max} (º)			
	Test 1	Test 2	Test 3	Average
Neoprene Rubber	45	46	46	46
36 Grit Sandpaper	45	48	50	48
80 Grit Sandpaper	47	46	45	46
150 Grit Sandpaper	44	45	43	44
220 Grit Sandpaper	41	42	42	42
360 Grit Sandpaper	36	35	37	36

2. Cutter Drag Force Test Results

Table 4: Cutter Drag Force Test Results

Test	Double Wall	Single Wall
	Peak Force (lbf)	Peak Force (lbf)
1	3.9	3.0
2	4.4	2.7

3	7.1	3.8
4	5.1	4.1
Average	5.1	3.4

3. Initial Prototype Stepper Motor Driver

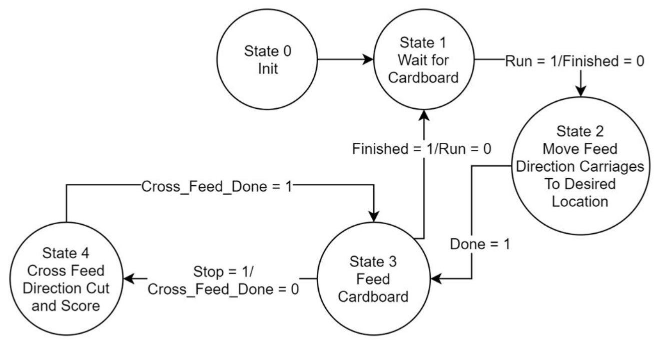


Figure 30: Initial Prototype Stepper Motor Driver State Transition Diagram

```python
# -*- coding: utf-8 -*-
"""
Jesus Valdez
Stepper Motor Driver
"""
import pyb
import time
import array as array
'''
----------------Stepper Motor Driver--------------------
pinE - Enable Pin (.high() to disable)
pinD - Direction Pin
pinP - Pulse Pin
SPR - Steps per revolution [step/rev] (Dont forget microstepping)
IPR - Inches per revolution [in/rev]
VMAX - Maximum velocity [in/s]
AMAX - Maximum acceleration [in/s^2]
MPP - Maximum Pulse Period [us] (Stepper Driver Limit)
MNPP - Minimum Pulse Width [us]
STEPS - Number of steps to take to full velocity(Step Resolution)
Hold - Power Stepper Always (May get Hot during long periods of use)
t_0 - First step pulse width
DPS - Direction Pin Switch (1 - Switch Direction)
'''
class stepper_motor_driver:

    def __init__(self,pinE,pinD,pinP,SPR,IPR,VMAX,AMAX,MPP,MNPP,STEPS,Hold,DPS=0):
        #-----------Declare Pins------------------
        self.Enable = pyb.Pin(pinE, pyb.Pin.OUT_PP)
        self.Direction = pyb.Pin(pinD, pyb.Pin.OUT_PP)
        self.Pulse = pyb.Pin(pinP, pyb.Pin.OUT_PP)
        #---Increasing Pulse Width Calculations---
        self.Steps_per_rev = SPR #[step/rev]
        self.Inches_per_rev = IPR #[in/rev]
        self.Max_velocity = VMAX #[in/s]
        self.Max_acceleration = AMAX #[in/s2]
        self.Maximum_pulse_width = MPP#[us]
        self.Minimum_pulse_width = MNPP#[us]
        self.Acceleration_steps = STEPS#[#]
        self.Pulse_width = array.array('I',[])
        self.Pulse_repeat = array.array('I',[])
        self.DPS = DPS
        self.state = 0
        self.toggle = 0
        #Check if Max_velocity is attainable and create Pulse Width Array
        self.Desired_Minimum_pulse_width = int(1000000*(self.Inches_per_rev/
          (self.Max_velocity*self.Steps_per_rev)))
        if self.Desired_Minimum_pulse_width>=self.Minimum_pulse_width:
            for n in range(0,(self.Acceleration_steps+1)):
```

```
                self.Pulse_width.append(int(self.Maximum_pulse_width-
                  ((self.Maximum_pulse_width-self.Desired_Minimum_pulse_width)/
                  self.Acceleration_steps)*n))
        elif self.Desired_Minimum_pulse_width<self.Minimum_pulse_width:
            for n in range(0,(self.Acceleration_steps+1)):
                self.Pulse_width.append(int(self.Maximum_pulse_width-
                  ((self.Maximum_pulse_width-self.Minimum_pulse_width)/
                  self.Acceleration_steps)*n))
        #Check if Max_acceleration is attainable and create Pulse Repeat Array
        self.Max_possible_acceleration = int((1/
          (self.Minimum_pulse_width*self.Minimum_pulse_width))*
          (self.Inches_per_rev/self.Steps_per_rev)*1000000*1000000)
        if self.Max_acceleration <= self.Max_possible_acceleration:
            for m in range(len(self.Pulse_width)-1):
                v_0 = (1/self.Pulse_width[m])*(self.Inches_per_rev/
                  self.Steps_per_rev)*1000000
                v_1 = (1/self.Pulse_width[(m+1)])*(self.Inches_per_rev/
                  self.Steps_per_rev)*1000000
                self.Pulse_repeat.append(int(((v_1-v_0)/self.Max_acceleration)/
                  (self.Pulse_width[m]*.000001)))
        elif self.Max_acceleration > self.Max_possible_acceleration:
            for m in range(len(self.Pulse_width)-1):
                v_0 = (1/self.Pulse_width[m])*(self.Inches_per_rev/
                  self.Steps_per_rev)*1000000
                v_1 = (1/self.Pulse_width[(m+1)])*(self.Inches_per_rev/
                  self.Steps_per_rev)*1000000
                self.Pulse_repeat.append(int(((v_1-v_0)/
                  self.Max_possible_acceleration)/(self.Pulse_width[m]*.000001)))
        self.number_of_steps = sum(self.Pulse_repeat) #Number of steps to max
          acceleration
        for n in range(len(self.Pulse_repeat)-1): #Convert steps to relative steps
                self.Pulse_repeat[n+1] = self.Pulse_repeat[n+1] + self.Pulse_repeat
                  [n]
        #Enable Stepper Motor Hold
        if Hold == True:
            self.Enable.low() #Motor Enable
        elif Hold == False:
            self.Enable.high() #Motor Disable

    #Toggles Pulse Pin
    def toggle_f(self):
        if self.toggle == 0:
            self.Pulse.high()
            self.toggle = 1
            return 1
        elif self.toggle == 1:
            self.Pulse.low()
            self.toggle = 0
            return 0
```

```
    #Measured move function
    '''
    Direction - "F" for Foward "R" for Reverse
    inches - length to move
    State - 0 Zero counters and change directions
            1 Move
            2 Pause
    Returns 3 when finished
    '''
    def measured_move(self,Direction,inches,State):
        #State 0 - Zero counters and set direction
        if (self.state == 0) and (State == 0):
            if self.DPS == 1:
                if Direction == "F":
                    self.Direction.low()
                elif Direction == "R":
                    self.Direction.high()
            elif self.DPS == 0:
                if Direction == "F":
                    self.Direction.high()
                elif Direction == "R":
                    self.Direction.low()
            self.current_location_s = 0
            self.Pulse_width_location = 0
            self.Pulse_repeat_location = 0
            self.last_recorded_time = time.ticks_us()
            self.required_number_steps = int((inches*self.Steps_per_rev)/
              self.Inches_per_rev)
            self.deccel_step=self.number_of_steps
            self.half_of_steps = self.required_number_steps//2
            self.half_step_deccel = self.half_of_steps

            #Check if full Acceleration and Deccelleration is posible
            if (self.number_of_steps*2) <= self.required_number_steps:
                self.full_accel_deccel=1
                self.state=1
            elif (self.number_of_steps*2) >= self.required_number_steps:
                self.full_accel_deccel=0
                self.state=1
            return 0

        #State 1 - Move
        #Full Acceleration & Deceleration
        if self.state == 1 and State == 1 and self.full_accel_deccel == 1 and
          self.deccel_step >= 1:
            #Accelerating State
            if self.current_location_s <= self.number_of_steps:
                if (time.ticks_us()-self.last_recorded_time) >= self.Pulse_width
```

```
                  [self.Pulse_width_location]:
                    self.toggle2 = self.toggle_f()
                    self.last_recorded_time = time.ticks_us()
                    if self.current_location_s == self.Pulse_repeat
                     [self.Pulse_repeat_location]:
                        self.Pulse_width_location = self.Pulse_width_location + 1
                        self.Pulse_repeat_location = self.Pulse_repeat_location + 1
                    self.current_location_s = self.current_location_s + 1
            #Not Accelerating State
            elif (self.current_location_s > self.number_of_steps):
                #Constant Velocity State
                if (self.required_number_steps-self.number_of_steps) >
                  self.current_location_s:
                    if (time.ticks_us()-self.last_recorded_time) >=
                     self.Pulse_width[self.Pulse_width_location-1]:
                        self.toggle2 = self.toggle_f()
                        self.last_recorded_time = time.ticks_us()
                        self.current_location_s = self.current_location_s + 1
                #Decceleration State
                elif (self.required_number_steps-self.number_of_steps) <=
                  self.current_location_s and (self.current_location_s <
                  self.required_number_steps):
                    if (time.ticks_us()-self.last_recorded_time) >=
                     self.Pulse_width[self.Pulse_width_location-1]:
                        self.toggle2 = self.toggle_f()
                        self.last_recorded_time = time.ticks_us()
                        if self.deccel_step == self.Pulse_repeat
                     [self.Pulse_repeat_location-1]:
                            self.Pulse_width_location = self.Pulse_width_location -
                      1
                            self.Pulse_repeat_location = self.Pulse_repeat_location
                      - 1
                        self.deccel_step = self.deccel_step - 1
                        self.current_location_s = self.current_location_s + 1
        #Partial Acceleration & Decceleration
        if self.state == 1 and State == 1 and self.full_accel_deccel == 0 and
          self.half_step_deccel >= 1:
            print("right location")
            #Accelerating State
            if self.current_location_s <= self.half_of_steps:
                if (time.ticks_us()-self.last_recorded_time) >= self.Pulse_width
                  [self.Pulse_width_location]:
                    self.last_recorded_time = time.ticks_us()
                    self.toggle2 = self.toggle_f()
                    if self.current_location_s == self.Pulse_repeat
                     [self.Pulse_repeat_location]:
                        self.Pulse_width_location = self.Pulse_width_location + 1
                        self.Pulse_repeat_location = self.Pulse_repeat_location + 1
                    self.current_location_s = self.current_location_s + 1
```

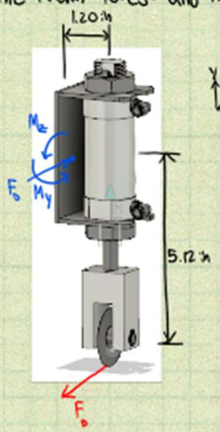
```
            #Deccelerating State
            elif self.current_location_s > self.half_of_steps:
                if (time.ticks_us()-self.last_recorded_time) >= self.Pulse_width 
                  [self.Pulse_width_location-1]:
                    self.toggle2 = self.toggle_f()
                    self.last_recorded_time = time.ticks_us()
                    if self.half_step_deccel == self.Pulse_repeat 
                     [self.Pulse_repeat_location-1]:
                            self.Pulse_width_location = self.Pulse_width_location - 
                      1
                            self.Pulse_repeat_location = self.Pulse_repeat_location 
                      - 1
                    self.half_step_deccel = self.half_step_deccel - 1
                    self.current_location_s = self.current_location_s + 1
        #Finished Moving
        if self.state == 1 and (self.deccel_step == 0 or self.half_step_deccel == 
          0):
            self.state = 0
            return 3
        #Still Moving
        if self.state == 1 and State == 1 and (self.deccel_step >= 1 or 
          self.half_step_deccel >= 1):
            return 1
        #Pause
        elif self.state == 1 and State == 2:
            return 2

    #constant_vel function
    '''
    Direction - "F" for Foward "R" for Reverse
    velocity - speed to move at
    State - 0 Hub State
            1 Move
    '''
    def constant_vel(self,Direction,velocity,State):
        #State 0 - zero variables find constant velocity pulse width
        if (State == 0):
            if self.DPS == 1:
                if Direction == "F":
                    self.Direction.low()
                elif Direction == "R":
                    self.Direction.high()
            elif self.DPS == 0:
                if Direction == "F":
                    self.Direction.high()
                elif Direction == "R":
                    self.Direction.low()
            self.constant_v_pulse = int((self.Inches_per_rev*1000000)/
              (velocity*self.Steps_per_rev))
```

```
            self.last_recorded_time = time.ticks_us()
        #State 1 - Begin moving at constant velocity
        if (State == 1) and (self.constant_v_pulse >= self.Minimum_pulse_width):
            if (time.ticks_us()-self.last_recorded_time)>=self.constant_v_pulse:
                self.toggle2 = self.toggle_f()
                self.last_recorded_time = time.ticks_us()
        elif (State == 1) and (self.constant_v_pulse < self.Minimum_pulse_width):
            if (time.ticks_us()-self.last_recorded_time)>=self.Minimum_pulse_width:
                self.toggle2 = self.toggle_f()
                self.last_recorded_time = time.ticks_us()

    #Returns location
    '''
    units - "S" for steps "In" for inches
    '''
    def location(self,units = "In"):
         if units == "S":
             return self.current_location_s
         elif units == "In":
             return int(self.current_location_s*(self.Inches_per_rev/
               self.Steps_per_rev))

    #Stepper Holding Torque
    '''
    hold = 1 to hold, hold = 0 to let go
    '''
    def hold(self,hold = 1):
        if hold == 1:
            self.Enable.low()
            return
        if hold == 0:
            self.Enable.high()
            return
```

4. Initial Protype Hand Calculations for Linear Bearings Selection

Sizing Linear Slider — Jesus Valtez

Using the experimentally found cutting drag force a linear slider can be chosen based on the known forces and moments.



Factor of Safety = 1.5

$F_D = (5.1\ lbf)\cdot(1.5) =$

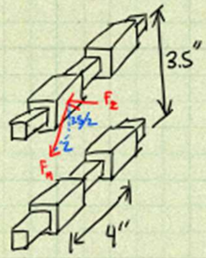
$F_D = 7.65\ lbf$

$M_z = (5.12\ in)\cdot(7.65\ ) = 39.2\ in\text{-}lb$

$M_y = (1.20\ in)\cdot(7.65\ ) = 9.18\ in\text{-}lb$

Assumption: Bearing friction and belt tension are negligible.

To eliminate moments 2 rails and 4 bearings will be used

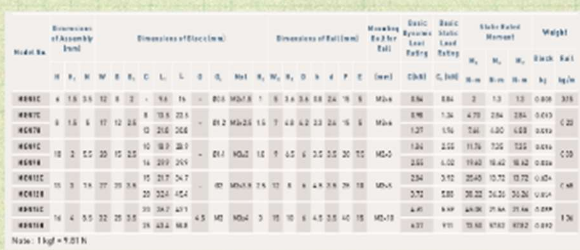


$F_z = \frac{9.18\ in\text{-}lb}{4\ in} = 2.295\ lb\ \ [.4019\ KN]$

$F_M = \frac{39.2\ in\text{-}lb}{\sqrt{(\frac{3.5}{2})^2 + 4^2}\ in} \times \frac{1}{4} = 3.687\ lb\ \ [0.646\ KN]$

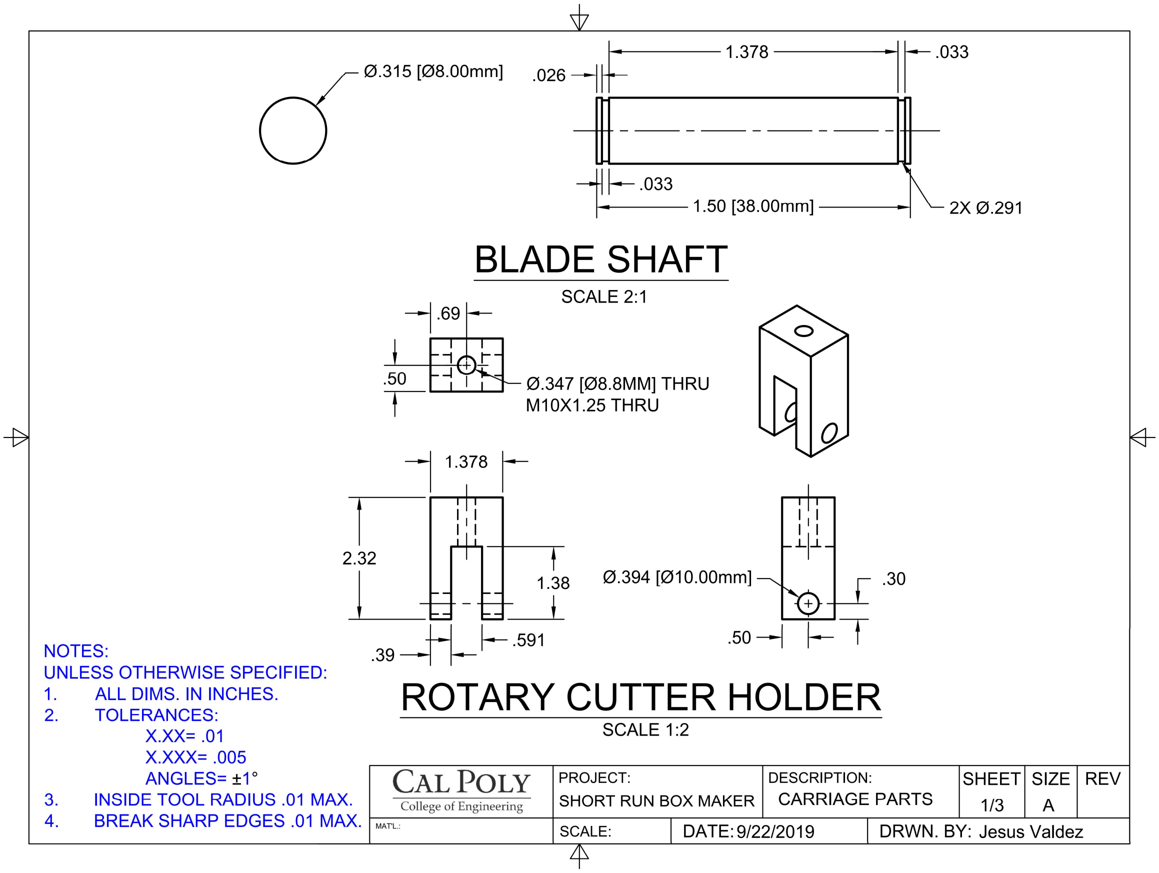
$F_{tot} = \sqrt{(.4019)^2 + (.646)^2} = 0.761\ KN$

Based on Calculations a MGN5C bearing from Hiwin meets the design criteria

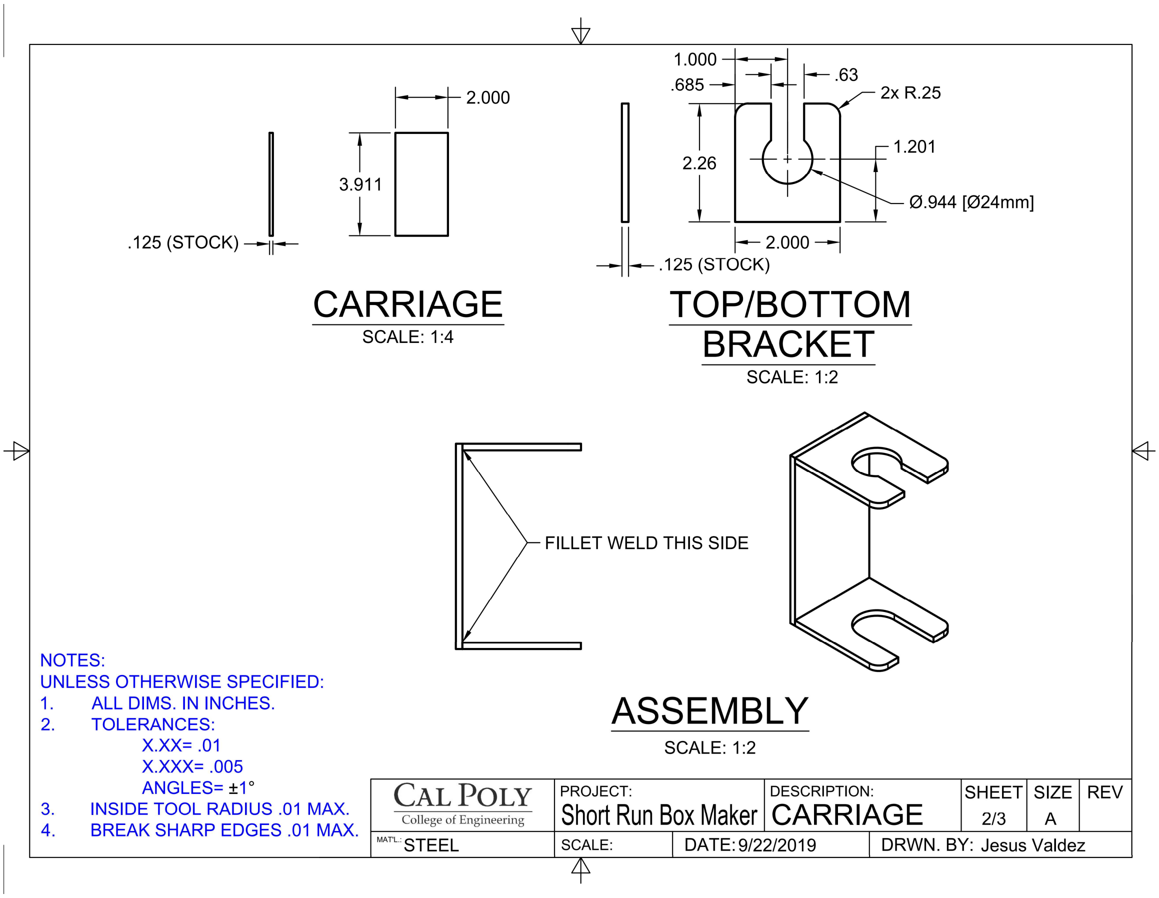


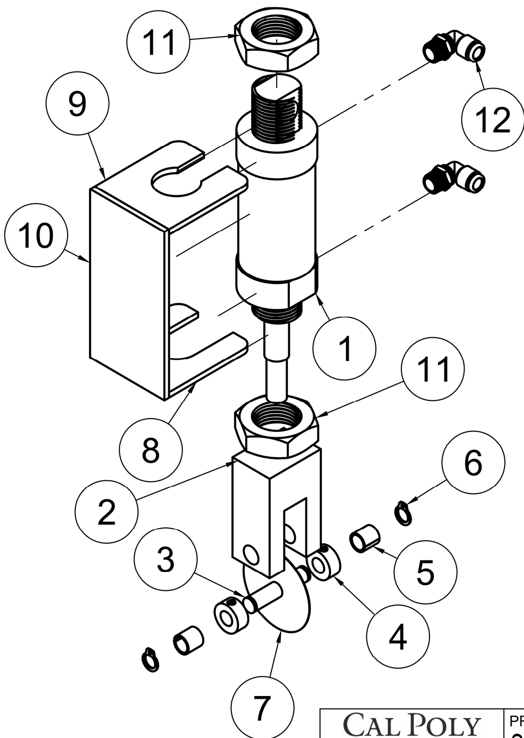
(HIWIN)

5. **Initial Prototype CAD Model of Carriage Assembly**



Ø.315 [Ø8.00mm]
.026
1.378
.033
.033
1.50 [38.00mm]
2X Ø.291
BLADE SHAFT
SCALE 2:1
.69
.50
Ø.347 [Ø8.8MM] THRU
M10X1.25 THRU
1.378
2.32
1.38
Ø.394 [Ø10.00mm]
.30
.39
.591
.50
ROTARY CUTTER HOLDER
SCALE 1:2
NOTES:
UNLESS OTHERWISE SPECIFIED:
1. ALL DIMS. IN INCHES.
2. TOLERANCES:
X.XX= .01
X.XXX= .005
ANGLES= ±1°
3. INSIDE TOOL RADIUS .01 MAX.
4. BREAK SHARP EDGES .01 MAX.
CAL POLY
College of Engineering
MAT'L:
PROJECT:
SHORT RUN BOX MAKER
DESCRIPTION:
CARRIAGE PARTS
SHEET
1/3
SIZE
A
REV
SCALE:
DATE: 9/22/2019
DRWN. BY: Jesus Valdez


2.000
3.911
.125 (STOCK)
CARRIAGE
SCALE: 1:4
1.000
.685
.63
2x R.25
1.201
2.26
Ø.944 [Ø24mm]
2.000
.125 (STOCK)
TOP/BOTTOM BRACKET
SCALE: 1:2
FILLET WELD THIS SIDE
ASSEMBLY
SCALE: 1:2
NOTES:
UNLESS OTHERWISE SPECIFIED:
1. ALL DIMS. IN INCHES.
2. TOLERANCES:
X.XX= .01
X.XXX= .005
ANGLES= ±1°
3. INSIDE TOOL RADIUS .01 MAX.
4. BREAK SHARP EDGES .01 MAX.
CAL POLY
College of Engineering
PROJECT:
Short Run Box Maker
DESCRIPTION:
CARRIAGE
SHEET
2/3
SIZE
A
REV
MAT'L: STEEL
SCALE:
DATE: 9/22/2019
DRWN. BY: Jesus Valdez



PARTS LIST			
ITEM	QTY	PART NUMBER	MATERIAL
1	1	MAL32X25 PNEUMATIC CYLINDER	
2	1	BLADE HOLDER	ALUMINUM
3	1	BLADE HOLDER SHAFT	STEEL
4	2	57485K67 SHAFT COLLAR	STEEL
5	2	6679K12 BRONZE BUSHING	BRONZE
6	2	98541A116 SNAP RING	STEEL
7	1	SOMOLUX45MM ROTARY CUTTER	STEEL
8	1	BOTTOM BRACKET	STEEL
9	1	TOP BRACKET	STEEL
10	1	CARRIAGE	STEEL
11	2	91938A175 M24-2.00 HEX NUT	STEEL
12	2	5225K722 PUSH-TO-CONNECT PIPE FITTING	STEEL

CAL POLY College of Engineering	PROJECT: Short Run Box Maker	DESCRIPTION: CARRIAGE ASSEMBLY	SHEET 3/3	SIZE A	REV
MAT'L.:	SCALE:1:2	DATE: 9/22/2019	DRWN. BY: Jesus Valdez		

6. Excel VBA UI Source Code

```
Sheet1 - 1

Dim Received_Data As String
Dim PortOpen As Integer
Dim BoxOption As Integer
Dim glueflap As Integer

'COM Port Send Button
Private Sub COMSend_Click()
    SP.Output = COMTextSend.Text
    COMTextSend.Text = ""
    Application.Wait (Now + TimeValue("0:00:2"))
    ComTextReceive.Text = ComTextReceive.Text + SP.InputData
End Sub

'Connect/Close Button
Private Sub ConnectButton_Click()
    ComTextReceive.Text = "Connecting"
    'If no COM port open
    If SP.PortOpen = False Then
        SP.PortOpen = True
        SP.CommPort = Range("M5").Value
        SP.Settings = "9600, N, 8, 1"
        ConnectButton.Caption = "Close"
        Application.Wait (Now + TimeValue("0:00:3"))
        ComTextReceive.Text = SP.InputData
        PortOpen = 1
    'If COM port open
    ElseIf SP.PortOpen = True Then
        SP.PortOpen = False
        ConnectButton.Caption = "Connect"
        ComTextReceive.Text = " "
        PortOpen = 0
    End If
End Sub


Private Sub glueflapcheckbox_Click()

End Sub

Private Sub OptionButton1_Click()
    'Box Pattern Option
    BoxOption = 0
End Sub

Private Sub OptionButton2_Click()
    'Half Box Pattern Option
    BoxOption = 1
End Sub

Private Sub OptionButton3_Click()
    '2-Flap Square Option
    BoxOption = 2
End Sub

Private Sub OptionButton4_Click()
    'Square Option
    BoxOption = 3
End Sub
'Send To SRBM Button
Private Sub SendSRBM_Click()
    If glueflapcheckbox.Value = True Then
    glueflap = 1
    ElseIf glueflapcheckbox.Value = False Then
    glueflap = 0
    End If
    ComTextReceive.Text = ComTextReceive.Text + ("Sending Instructions..." & vbCrLf)
    Application.Wait (Now + TimeValue("0:00:1"))
    ComTextReceive.Text = ComTextReceive.Text + CStr(Range("B20").Value) + "," + CStr(Range("C20").Val
ue) + "," + CStr(Range("D20").Value) + "," + CStr(Range("E20").Value) + "," + CStr(Range("C23").Value)
 + "," + CStr(glueflap) + "," + CStr(BoxOption) + ("," & vbCrLf)
    SP.Output = CStr(Range("B20").Value) + "," + CStr(Range("C20").Value) + "," + CStr(Range("D20").Va
lue) + "," + CStr(Range("E20").Value) + "," + CStr(Range("C23").Value) + "," + CStr(glueflap) + "," +
CStr(BoxOption) + ","
```

```
    Application.Wait (Now + TimeValue("0:00:3"))
    ComTextReceive.Text = ComTextReceive.Text + ("Instructions Succesfully Sent" & vbCrLf)
    Application.Wait (Now + TimeValue("0:00:3"))
    ComTextReceive.Text = SP.InputData
End Sub

Private Sub Worksheet_SelectionChange(ByVal Target As Range)
End Sub
```

7. ISR Hand Calculations

ISR Hand Calculations | Jesus Valdez

Calculating the Steps per inch for each axis.

Y-axis

$$200\ \text{Pulse/rev} \times 20\ \text{Teeth Pulley} \times \frac{1\ T}{.0787\ \text{in}} \times 16\ \text{microsteps} = \boxed{2033\ \text{steps/in}}$$

(Belt Pitch: $\frac{1\ T}{.0787\ \text{in}}$)

X-axis

$$200\ \text{Pulse/rev} \times \left(\frac{1}{3.14 \times 1.0625\ \text{in}}\right) \times \left(\frac{13T}{6T}\right) \times 16\ \text{microsteps} = \boxed{2078\ \text{steps/in}}$$

(Roller Circumference: 3.14×1.0625 in)

Calculating max velocity number of return to interrupt (rti)

Y-axis

$$\frac{100{,}000\ \text{HZ}}{(1\ \text{in/s})(2033\ \text{steps/in})} \approx \boxed{49\ \text{rti}}$$

(100,000 HZ = Interrupt Freq.; 1 in/s = max. vel.)

X-axis

$$\frac{100{,}000\ \text{HZ}}{(1\ \text{in/s})(2078)} \approx \boxed{48\ \text{rti}}$$

(100,000 HZ = Interrupt freq.; 1 in/s = max. vel)

8. SRBM Final Prototype Source Code

Files:

SRBM.ino

Stepper.h

Stepper.cpp

Serial.h

Serial.cpp

UI.h

UI.cpp

Limit_Switches.h

Limit_Switches.cpp

SRBM.ino

```
#include "Stepper.h"
#include "Limit_Switches.h"
#include "UI.h"
#include "Serial.h"


void setup()
{
    stepper_task();
    limit_switch_init();
    UI_init();
    serial_task();
}

void loop()
{
    stepper_task();
    serial_task();
}

```

Stepper.h

```
#ifndef STEPPER_H_
#define STEPPER_H_

// Stepper Pulse Pin Mask
#define pulse_mask B11110000;
// Stepper Direction Pin Mask
#define dir_mask B00000111;
//PortD
#define x_axis_pulse_mask B00010000;
#define y_axis_pulse_mask B00100000;
#define l_pulse_mask B10000000;
//PortB
#define x_axis_dir_mask B00000001;
#define y_axis_dir_mask B00000010;

void stepper_task();
void movement_subtask();
extern int instructions_received_flag;
#endif /* STEPPER_H_ */

```

Page 1

```
#include "Arduino.h"
#include "Stepper.h"
#include "Serial.h"
#include "UI.h"
#include "Limit_Switches.h"

/*Stepper Task

Controls Stepper Motor and Laser Output via an ISR (Interrupt Service Routine)
Instruction are loaded into coordinates[40][2] and mode[40] in Serial.cpp
When instructions_received_flag is set high the following is done
1. Y Axis Home (Laser Mirror Axis)
2. Waits for Play Button to be pressed
2. X Axis Home
3. Waits for Play Button to be pressed
4. Executes box instructions
5. return to Wait state

Additional features
-While in Wait State the push out button moves the x axis to remove remaining cardboard in the rollers
-If instructions have been loaded the Play button will repeat instructions

*/
//Shared Variables
int instructions_received_flag = 0;//Flag Set when instructions received
//Movement Calculation Variables
//
volatile int Number_of_acc_steps_x = 10;//Number of Steps to Max Velocity
volatile int Number_of_acc_steps_y = 10;//Number of Steps to Max Velocity
int Number_of_acc_steps_x2;//Number of Steps to Max Velocity
int steps_per_inch_x = 200*16*(1/20.0)*(1/.0787);//(200 p/rev)*(16microsteps)*(1/20 Pulley Teeth)*(1T/.0787in Belt Pitch)
int steps_per_inch_y = 200*16*(1/(3.14*1.0625))*(13/6);//(200 p/rev)*(16microsteps)*(1 / (pi*1.0625 in Roller diam.))*(13T/6T Spur Gear Teeth)
volatile int x_vel_rti,x_rti_array[10];
volatile int y_vel_rti,y_rti_array[10];
volatile int x_vel_act_rti, y_vel_act_rti;
volatile long x_number_steps, y_number_steps, x_location, y_location, x_last_rti, y_last_rti, l_last_rti=0;
volatile int xmove = 0, ymove = 0, lmove = 0,l_pattern,l_toggle = 0;
// Set Equal to -1 to flip direction
int x_flip_direction=1;
int y_flip_direction=1;

//Tested Interrupt Fq.
unsigned long interrupt_fq = 100000;
volatile int const_move = 0;
volatile int const_move_toggle = 0;
volatile int const_move_counter = 0;
void stepper_task()
{
    static int State = 0;
    //State 0 - Init.
    if (State == 0) {
        //Set Outputs
        DDRD = DDRD|pulse_mask;//Pulse Pins
        DDRB = DDRB|dir_mask;//Direction Pins
        delay(1000);
```

```
        PORTD = B10000000;//All Port D Low Set Laser Output High because Low Active

        //Enable Timer Compare Interrupts
        //Clear Timer Register
        TCCR1A = 0;
        TCCR1B = 0;
        TCNT1 = 0;

        OCR1A = 5; // compare match register
        TCCR1B |= (1<<WGM12); // CTC mode
        TCCR1B |= (1<<CS11); // 8 prescaler
        State = 1;
        return;
    }
    //State 1 - Wait
    if (State == 1){
        if (instructions_received_flag == 1){
            State = 2;
        }
        UI_task();
        if(ui_push_out==1){
            const_move = 2;
            TIMSK1 = TIMSK1 | B00000010;//Enable Interrupts
            PORTB = PORTB | y_axis_dir_mask;//Direction to Home
            while(ui_push_out==1){
                UI_task();
            }
            TIMSK1 = TIMSK1 ^ B00000010;//Disable Interrupts
            const_move = 0;
        }
        if(memory_flag==1 && ui_repeat == 1){
            State = 2;
        }
        return;
    }
    //State 2 - Home Cycle
    if (State == 2){
        movement_subtask();
        State = 3;
        return;
    }
    //State 3 - Process Movement
    if (State == 3){
        movement_subtask();
        State = 4;
        return;
    }
    //State 4 - Go To Exit Position
    if (State == 4){
        State = 1;
        return;
    }
}
//Movement Subtask
void movement_subtask()
{
    static int State = 0;
    //State 0 - Home Cycle
    if (State == 0) {
```

```
        //Retrieve Stepper Info from Serial Buffer
        x_vel_rti = interrupt_fq/(velocity*steps_per_inch_x);
        y_vel_rti = interrupt_fq/(velocity*steps_per_inch_y);

        for(int n = 0;n <= 9;n++){
            x_rti_array[n]=(10/(1.0+n))*x_vel_rti;
            y_rti_array[n]=(10/(1.0+n))*y_vel_rti;
        }
        l_pattern = .5*steps_per_inch_x;//inches of pattern * steps per inch * rti
        State = 1;
        //Home Y Axis--------------------------------
        const_move = 1;//Y axis
        const_move_counter=0;
        TIMSK1 = TIMSK1 | B00000010;//Enable Interrupt
        PORTB = PORTB | y_axis_dir_mask;//Direction to Home
        limit_x =0;
        while(limit_x == 0){
            limit_switch_task();//Check when limit has been hit
        }
        TIMSK1 = TIMSK1 ^ B00000010;//Disable Interrupt
        const_move = 0;
        return;
    }
    //State 1 - Process Movement
    //
    if (State == 1){
        //Wait for Play Button pressed - Then Load Sheet
        ui_play=0;
        //Loop Stepper Motor until Limit Switch Indicates Sheet of Cardboard is loaded
        while(ui_play==0){
            UI_task();
        }
        const_move = 2;//X axis
        const_move_counter=0;
        //Home X Axis--------------------------------
        TIMSK1 = TIMSK1 | B00000010;//Enable Interrupt
        PORTB = PORTB | x_axis_dir_mask;//Direction to Home
        limit_board = 0;
        while(limit_board == 0){
            limit_switch_task();//Check when limit has been hit
        }
        TIMSK1 = TIMSK1 ^ B00000010;//Disable Interrupt
        const_move = 0;
        //Wait for Play Button - Then Begin Cutting
        ui_play=0;
        while(ui_play==0){
            UI_task();
        }
        for (int index = 0;index < number_of_instructions;index++){
            x_number_steps = abs(coordinates[index][0]*steps_per_inch_x);
            y_number_steps = abs(coordinates[index][1]*steps_per_inch_y);
            //Check Direction pins
            if((coordinates[index][0]*x_flip_direction)>=0){
                PORTB = PORTB | x_axis_dir_mask;
            }
            else{
                PORTB = PORTB ^ x_axis_dir_mask;
            }

```

```
        if((coordinates[index][1]*y_flip_direction)>=0){
            PORTB = PORTB | y_axis_dir_mask;
        }
        else{
            PORTB = PORTB ^ y_axis_dir_mask;
        }
        xmove = 1;
        ymove = 1;
        lmove = 0;
        if(x_number_steps == 0){
            xmove = 0;
        }
        if(y_number_steps == 0){
            ymove = 0;
        }
        //Enable Cut Mode
        if(mode[index]==1){
            PORTD = PORTD ^ l_pulse_mask;
        }
        if(mode[index]==2){
            lmove = 1;
        }
        x_location = 0;
        y_location = 0;
        x_vel_act_rti=x_rti_array[0];
        y_vel_act_rti=y_rti_array[0];
        x_last_rti = 0;
        y_last_rti = 0;
        TIMSK1 = TIMSK1 | B00000010;//Enable Timer Interrupts
        while((xmove == 1)|(ymove == 1)){
            //Loop while ISR moves steppers
            //Wait for finished movement
            //Checks for Reset Buttons
            UI_task();
            if(ui_reset == 1){
                xmove = 0;
                ymove = 0;
                ui_play = 0;
                ui_pause = 0;
                ui_push_out = 0;
                ui_reset = 0;
                ui_repeat=0;
                limit_x = 0;
                limit_x2 = 0;
                limit_board = 0;
                break;
            }
        }
        TIMSK1 = TIMSK1 ^ B00000010;//Disable timer compare interrupt
        PORTD = B00100000;
        PORTD = PORTD | l_pulse_mask;//Disable Laser Active Low
        lmove = 0;
        l_last_rti=0;
        xmove = 0;
        ymove = 0;
    }
    State = 0;
    instructions_received_flag = 0;
    ui_play = 0;
```

```
        ui_pause = 0;
        ui_push_out = 0;
        ui_reset = 0;
        ui_repeat=0;
        limit_x = 0;
        limit_x2 = 0;
        limit_board = 0;
        return;

    }
}
//Interrupt Service Routine
ISR(TIMER1_COMPA_vect){
    if(ui_pause == 0){
    //X Axis Calculations
    if(xmove == 1){
        x_last_rti++;
        if (x_last_rti == x_vel_act_rti){
            l_last_rti++;
            PORTD = PORTD | x_axis_pulse_mask;
            x_last_rti = 0;
            x_location++;
            x_vel_act_rti = x_vel_rti;
            if(x_location <= Number_of_acc_steps_x){
                x_vel_act_rti =x_rti_array[x_location];
            }
            if(x_location > (x_number_steps-Number_of_acc_steps_x)){
                x_vel_act_rti =x_rti_array[x_number_steps-x_location];
            }
            if(x_location == x_number_steps){
                xmove = 0;
            }
        }
        //Toggle Pin off
        if ((x_last_rti == 20)){
            PORTD = PORTD ^ x_axis_pulse_mask;
        }
    }
    //Y Axis Calculations
    if(ymove == 1){
        y_last_rti++;
        if (y_last_rti == y_vel_act_rti){
            l_last_rti++;
            PORTD = PORTD | y_axis_pulse_mask;
            y_last_rti = 0;
            y_location++;
            y_vel_act_rti = y_vel_rti;
            if(y_location <= Number_of_acc_steps_y){
                y_vel_act_rti =y_rti_array[y_location];
            }
            if(y_location > (y_number_steps-Number_of_acc_steps_y)){
                y_vel_act_rti =y_rti_array[y_number_steps-y_location];
            }
            if(y_location == y_number_steps){
                ymove = 0;
            }
        }
                //Toggle Pin off
        if ((y_last_rti == 20)){
```

```
            PORTD = PORTD ^ y_axis_pulse_mask;
        }

    }
    //Laser Pulse Calculation
    if(lmove == 1){
        if(l_last_rti == l_pattern ){
            PORTD = PORTD | l_pulse_mask;

        }
        if(l_last_rti == l_pattern*2 ){
            PORTD = PORTD ^ l_pulse_mask;
            l_last_rti = 0;
        }
    }
    //Turn off laser during pause
    if(ui_pause == 1){
        PORTD = PORTD | l_pulse_mask;//Active Low
    }
    }
    //Home Cycles
    if(const_move == 1 || const_move == 2){
        //X axis
        if(const_move==2){
            const_move_counter++;
            if(const_move_counter == 50){
                PORTD = PORTD | x_axis_pulse_mask;

            }
            if(const_move_counter == 50){
                const_move_counter=0;
                PORTD = PORTD ^ x_axis_pulse_mask;
            }
        }
        //Y axis
        if(const_move==1){
            const_move_counter++;
            if(const_move_counter == 50){
                PORTD = PORTD | y_axis_pulse_mask;

            }
            if(const_move_counter == 50){
                const_move_counter=0;
                PORTD = PORTD ^ y_axis_pulse_mask;
            }
        }
    }
};

```

Serial.h

```
#ifndef SERIAL_H_
#define SERIAL_H_
void serial_task();
void coordinate_array();
extern float length,width,height,flap_length;
extern int velocity,joint_tab,type;
extern float coordinates[40][2];
extern int mode[40];
extern int number_of_instructions;
extern int memory_flag;
#endif /* SERIAL_H_ */
```

```
#include "Arduino.h"
#include "Serial.h"
#include "Stepper.h"

/* Serial Task
 *
 * Waits for incoming instructions from Serial which it then converts into
 * a list of instructions for the Stepper Driver
 *
 * Incoming Serial format
 * "L,W,H,F,V,J,T"
 * L - length [in]
 * W - width [in]
 * H - height [in]
 * F - Flap length [in]
 * V - Velocity [in/s]
 * J - Joint Tab [Boolean]
 * T - Type of Box [int]
 *      0 - Box Pattern
 *      1 - Half Box pattern
 *      2 - Double Flap Square
 *      3 - Square
 *
 * Additional instructions can be added to the coordinate_array() function
 */

//Shared Variables
String Serial_received;
float length,width,height,flap_length;
int velocity,joint_tab,type;
float coordinates [40][2] = {};
int mode[40] = {};
int number_of_instructions;
int memory_flag=0;//Set to 1 f instructions buffer is loaded with instructions
//This allows repeat of instructions
void serial_task()
{
    static int State = 0;

    //State 0 - Init.
    if (State == 0) {
        Serial.begin(9600);
        //Wait for Serial Connection
        while (!Serial) {
        ;
        }
        Serial.print("Short Run Box Maker Connected Successfully");
        State = 1;
        return;
    }
    //State 1 - Wait for Transmission
    if (State == 1){
        //Set Instruction Received Flag
        instructions_received_flag = 0;
        //Wait for Transmission
        if (Serial.available() > 0){
            State = 2;
        }
        return;
```

```
    }
    //State 2 - Store Transmission
    if (State == 2){
        Serial_received = Serial.readStringUntil(',');
        length = Serial_received.toFloat();
        Serial.read();
        Serial_received = Serial.readStringUntil(',');
        width = Serial_received.toFloat();
        Serial.read();
        Serial_received = Serial.readStringUntil(',');
        height = Serial_received.toFloat();
        Serial.read();
        Serial_received = Serial.readStringUntil(',');
        flap_length = Serial_received.toFloat();
        Serial.read();
        Serial_received = Serial.readStringUntil(',');
        velocity = Serial_received.toInt();
        Serial.read();
        Serial_received = Serial.readStringUntil(',');
        joint_tab = Serial_received.toInt();
        Serial.read();
        Serial_received = Serial.readStringUntil(',');
        type = Serial_received.toInt();
        Serial.read();
        coordinate_array();
        State = 1;
        //Set Instruction Received Flag
        instructions_received_flag = 1;
        memory_flag = 1;
        return;
    }
}

//Function That Stores Box Coordinates
void coordinate_array(){
    //Box Pattern
    if (type == 0) {
        if (joint_tab == 1){
            float coordinates2[40][2] = {{0,flap_length}\
            ,{length,0},{0,-flap_length},{-length,0},{0,(2*flap_length+height)},{length,0},{0,-flap_length},{-length,0},{length,0},{0,-height}\
            ,{width,0},{0,-flap_length},{-width,0},{0,(2*flap_length+height)},{width,0},{0,-flap_length},{-width,0},{width,0},{0,-height}\
            ,{length,0},{0,-flap_length},{-length,0},{0,(2*flap_length+height)},{length,0},{0,-flap_length},{-length,0},{length,0},{0,-height}\
            ,{width,0},{0,-flap_length},{-width,0},{0,(2*flap_length+height)},{width,0},{0,-flap_length},{-width,0},{width,0},{0,-height}\
            ,{1,0},{0,height},{-1,0}};
            int mode2[] = {0,2,1,0,0,1,1,2,0,2\
            ,2,1,0,0,1,1,2,0,2\
            ,2,1,0,0,1,1,2,0,2\
            ,2,1,0,0,1,1,2,0,2\
            ,1,1,1};
            number_of_instructions = 40;
            for(int n =0;n<=number_of_instructions; n++){
                            coordinates[n][0]=coordinates2[n][0];
                            coordinates[n][1]=coordinates2[n][1];
                            mode[n]=mode2[n];
                    }
```

```
        }
        if (joint_tab == 0){
            float coordinates2[37][2] = {{0,flap_length}\
            ,{length,0},{0,-flap_length},{-length,0},{0,(2*flap_length+height)},{length,0},{0,-flap_length},{-length,0},{length,0},{0,-height}\
            ,{width,0},{0,-flap_length},{-width,0},{0,(2*flap_length+height)},{width,0},{0,-flap_length},{-width,0},{width,0},{0,-height}\
            ,{length,0},{0,-flap_length},{-length,0},{0,(2*flap_length+height)},{length,0},{0,-flap_length},{-length,0},{length,0},{0,-height}\
            ,{width,0},{0,-flap_length},{-width,0},{0,(2*flap_length+height)},{width,0},{0,-flap_length},{-width,0},{width,0},{0,-height}};
            int mode2[] = {0,2,1,0,0,1,1,2,0,2\
                        ,2,1,0,0,1,1,2,0,2\
                        ,2,1,0,0,1,1,2,0,2\
                        ,2,1,0,0,1,1,2,0,2};
            number_of_instructions = 37;
            for(int n =0;n<=number_of_instructions; n++){
                        coordinates[n][0]=coordinates2[n][0];
                        coordinates[n][1]=coordinates2[n][1];
                        mode[n]=mode2[n];
                }
        }
    }
    //Half Box Pattern
    if (type == 1) {
        if (joint_tab == 1){
            float coordinates2[22][2] = {{0,flap_length}\
            ,{length,0},{0,-flap_length},{-length,0},{0,(2*flap_length+height)},{length,0},{0,-flap_length},{-length,0},{length,0},{0,-height}\
            ,{width,0},{0,-flap_length},{-width,0},{0,(2*flap_length+height)},{width,0},{0,-flap_length},{-width,0},{width,0},{0,-height}\
            ,{1,0},{0,height},{-1,0}};
            int mode2[] = {0,2,1,0,0,1,1,2,0,2\
            ,2,1,0,0,1,1,2,0,2\
            ,1,1,1};
            number_of_instructions = 22;
            for(int n =0;n<=number_of_instructions; n++){
                        coordinates[n][0]=coordinates2[n][0];
                        coordinates[n][1]=coordinates2[n][1];
                        mode[n]=mode2[n];
                }
        }
        if (joint_tab == 0){
            float coordinates2[19][2] = {{0,flap_length}\
            ,{length,0},{0,-flap_length},{-length,0},{0,(2*flap_length+height)},{length,0},{0,-flap_length},{-length,0},{length,0},{0,-height}\
            ,{width,0},{0,-flap_length},{-width,0},{0,(2*flap_length+height)},{width,0},{0,-flap_length},{-width,0},{width,0},{0,-height}};
            int mode2[] = {0,2,1,0,0,1,1,2,0,2\
                        ,2,1,0,0,1,1,2,0,2};
            number_of_instructions = 19;
            for(int n =0;n<=number_of_instructions; n++){
                        coordinates[n][0]=coordinates2[n][0];
                        coordinates[n][1]=coordinates2[n][1];
                        mode[n]=mode2[n];
                }
        }
    }
    //2 - Flap Square
```

```
    if (type == 2) {
        float coordinates2[9][2] = {{0,height},{length,0},{0,-height}\
        ,{0,height},{width,0},{0,-height}\
        ,{0,height},{flap_length,0},{0,-height}};
        int mode2[] = {0,1,2,0,1,2,0,1,2};
        number_of_instructions = 9;
        for(int n =0;n<=number_of_instructions; n++){
                        coordinates[n][0]=coordinates2[n][0];
                        coordinates[n][1]=coordinates2[n][1];
                        mode[n]=mode2[n];
                }
    }
    //Square
    if (type == 3) {
        float coordinates2[3][2] = {{0,width},{length,0},{0,-width}};
        int mode2[] = {0,1,1};
        number_of_instructions = 3;
        for(int n =0;n<=number_of_instructions; n++){
                coordinates[n][0]=coordinates2[n][0];
                coordinates[n][1]=coordinates2[n][1];
                mode[n]=mode2[n];
        }

    }

}

```

```

#ifndef UI_H_
#define UI_H_
// User Switch Pins
#define UI_PINS B00001111
extern int ui_play;
extern int ui_pause;
extern int ui_push_out;
extern int ui_reset;
extern int ui_repeat;
void UI_init();
void UI_task();



#endif /* UI_H_ */

```

```
#include "Arduino.h"
#include "UI.h"
#include "Stepper.h"
int ui_play = 0;
int ui_pause = 0;
int ui_push_out = 0;
int ui_reset = 0;
int ui_repeat=0;

/* UI Task
 * The following task checks for button states
 * The function UI_task can be called whenever user input
 * is expected or to control movement
 */

void UI_init()
{
    //Set Pins as inputs
    DDRC = DDRC^UI_PINS;

}
void UI_task()
{
        //Play Button - A0
        if(((PINC&B00000001)==B00000001) && ui_play == 0){
            ui_play = 1;
            ui_pause = 0;
            ui_repeat = 1;
            return;
        }
        ui_repeat = 0;
        // Pause Button - A1
        if(((PINC&B00000010)==B00000010) && ui_pause == 0){
            ui_pause = 1;
            ui_play = 0;
            return;
        }
        //Push Out - A2
        if(((PINC&B00000100)==B00000100) && ui_push_out == 0){
            ui_push_out = 1;
            return;
        }
        ui_push_out = 0;
        //Reset - A3
        if(((PINC&B00001000)==B00001000) && ui_reset == 0){
            ui_reset = 1;
            return;
        }
        ui_reset = 0;
        return;
}
```

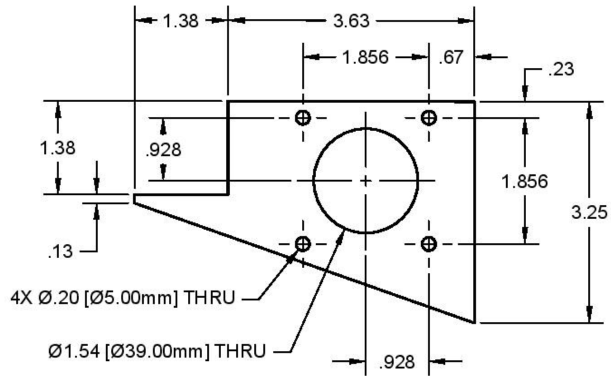
```
#ifndef LIMIT_SWITCHES_H_
#define LIMIT_SWITCHES_H_
// Limit Switches Pins
#define LIMIT_SWITCH_PINS B00111000;
extern int limit_x;
extern int limit_x2;
extern int limit_board;
void limit_switch_init();
void limit_switch_task();


#endif /* LIMIT_SWITCHES_H_ */
```

Limit_Switches.cpp

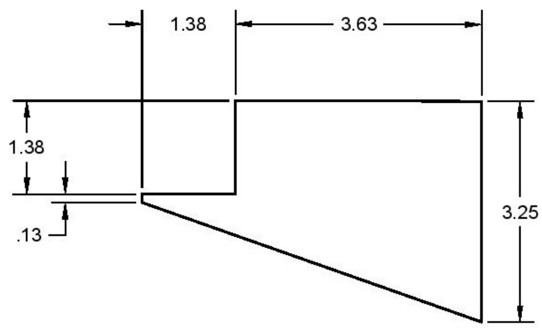
```cpp
#include "Limit_Switches.h"
#include "Arduino.h"
int limit_x = 0;
int limit_x2 = 0;
int limit_board = 0;


/* Limit Switch Task
 * The following task checks the state of the limit switches
 * an additional pin was coded in as x2 for an additional limit switch
 * that may be needed for further development
 * The function limit_switch_task can be called whenever limit switch
 * state is required
 *
 */
void limit_switch_init()
{
    //Set Pins as Inputs
    DDRB = DDRB^LIMIT_SWITCH_PINS;
}

void limit_switch_task()
{

    //Limit Board - Pin 13
    if((PINB&B00100000)==B00100000){
        limit_board = 1;
        return;
    }
    limit_board = 0;
    //Limit X2 - Pin 12
    if((PINB&B00010000)==B00010000){
        limit_x2 = 1;
        return;
    }
    limit_x2 = 0;
    //Limit X - Pin 11
    if((PINB&B00001000)==B00001000){
        limit_x = 1;
        return;
    }
    limit_x = 0;

}

```

Page 1

9. Final Prototype CAD



Part 1 - Bottom Mount

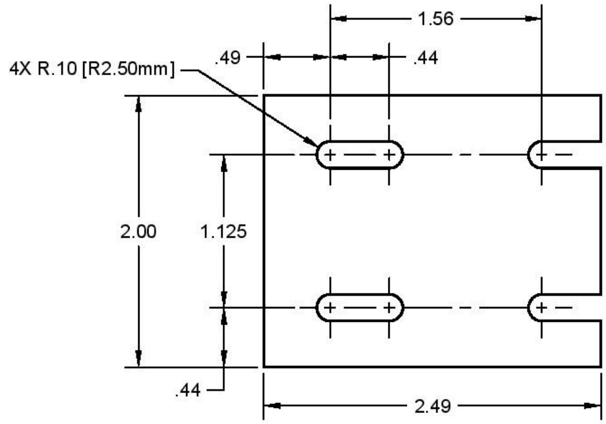


Part 2 - Top Mount

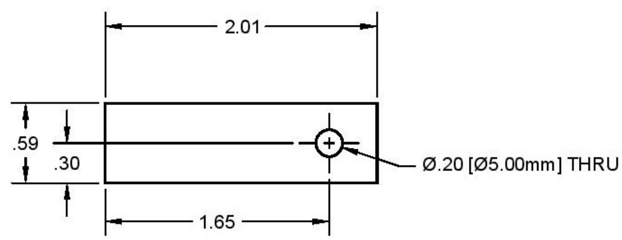
NOTES:
MATERIAL: 14G HOT ROLLED STEEL
UNLESS OTHERWISE SPECIFIED:

1. ALL DIMS. IN INCHES
2. TOLERANCES
   X.XX = ±.01
   X.XXX = ±.005
   ANGLES = ±1°
3. INSIDE TOOL RADIUS .01 MAX.
4. BREAK SHARP EDGES .01 MAX.

APPROVED	SIZE	CODE	DWG NO	TITLE Short Run Box Maker	REV
CHECKED	A				
DRAWN Jesus Valdez 6/2/2020	SCALE 1:10		WEIGHT	SHEET 1/14	



Part 3 - Mirror Mount

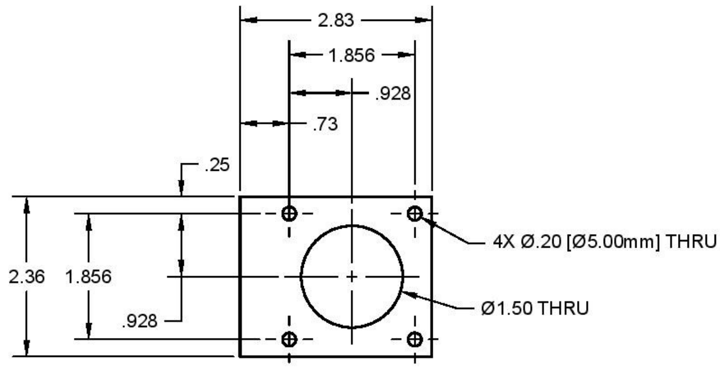


Part 4 - X AXIS Pulley Mount

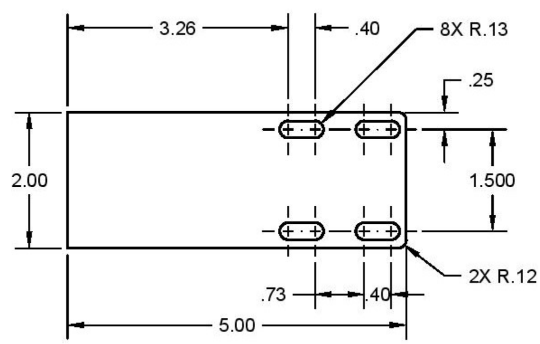
NOTES:
MATERIAL: 14G HOT ROLLED STEEL
UNLESS OTHERWISE SPECIFIED:

1. ALL DIMS. IN INCHES
2. TOLERANCES
   X.XX = ±.01
   X.XXX = ±.005
   ANGLES = ±1°
3. INSIDE TOOL RADIUS .01 MAX.
4. BREAK SHARP EDGES .01 MAX.

APPROVED	SIZE	CODE	DWG NO	TITLE	REV
CHECKED	A			Short Run Box Maker	
DRAWN Jesus Valdez 6/2/2020	SCALE 1:10		WEIGHT	SHEET 2/14	



PART 5 - X MOTOR MOUNT

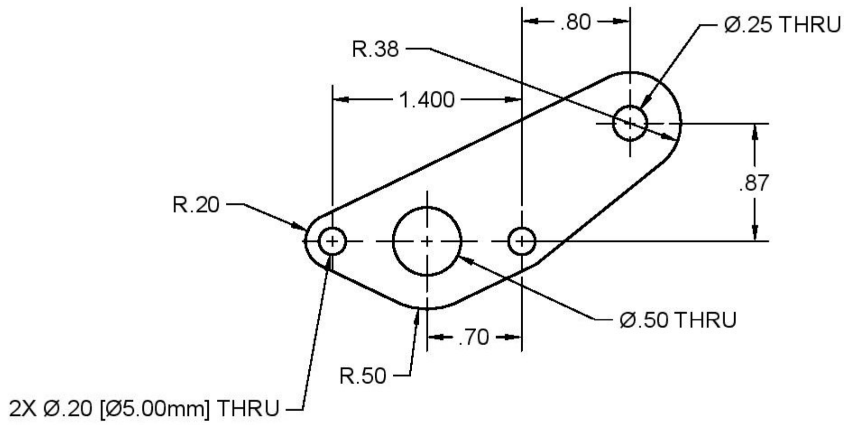


PART 6 - LASER FRAME BRACKET

NOTES:
MATERIAL: 14G HOT ROLLED STEEL
UNLESS OTHERWISE SPECIFIED:

1. ALL DIMS. IN INCHES
2. TOLERANCES
   X.XX = ±.01
   X.XXX = ±.005
   ANGLES = ±1°
3. INSIDE TOOL RADIUS .01 MAX.
4. BREAK SHARP EDGES .01 MAX.

APPROVED	SIZE	CODE	DWG NO	TITLE Short Run Box Maker	REV
CHECKED	A				
DRAWN Jesus Valdez 6/2/2020	SCALE 1:30		WEIGHT	SHEET 3/14	

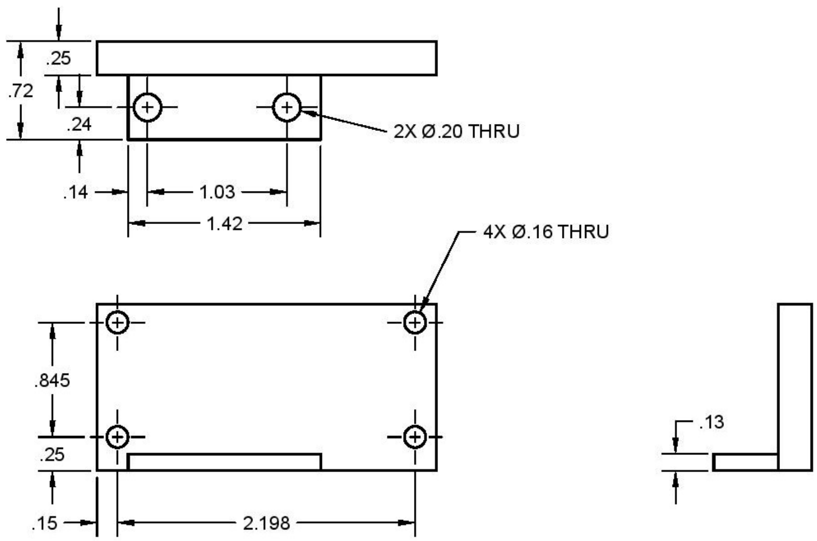


PART 7 - TOP ROLLER BRACKET

NOTES:
MATERIAL: 14G HOT ROLLED STEEL
UNLESS OTHERWISE SPECIFIED:

1. ALL DIMS. IN INCHES
2. TOLERANCES
   - X.XX = ±.01
   - X.XXX = ±.005
   - ANGLES = ±1°
3. INSIDE TOOL RADIUS .01 MAX.
4. BREAK SHARP EDGES .01 MAX.

APPROVED	SIZE	CODE	DWG NO	TITLE Short Run Box Maker Final Design_Top Roller Bracket	REV
CHECKED	A				
DRAWN Jesus Valdez 6/9/2020	SCALE 1:1		WEIGHT	SHEET 1/1	

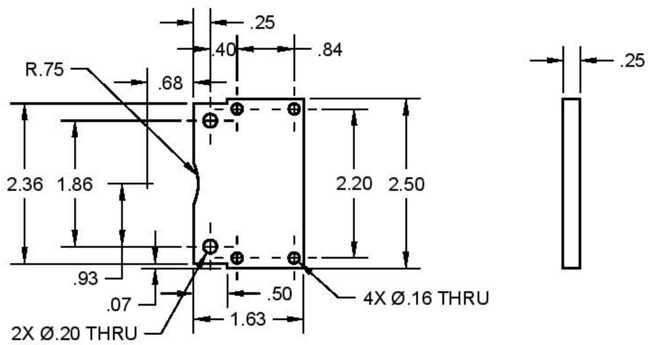


PART 8 - Y LIMIT SWITCH MOUNT

NOTES:
MATERIAL:
PART 7: 14G HOT ROLLED STEEL
PART 8: PLA PLASTIC
UNLESS OTHERWISE SPECIFIED:

1. ALL DIMS. IN INCHES
2. TOLERANCES
   X.XX = ±.01
   X.XXX = ±.005
   ANGLES = ±1°
3. INSIDE TOOL RADIUS .01 MAX.
4. BREAK SHARP EDGES .01 MAX.

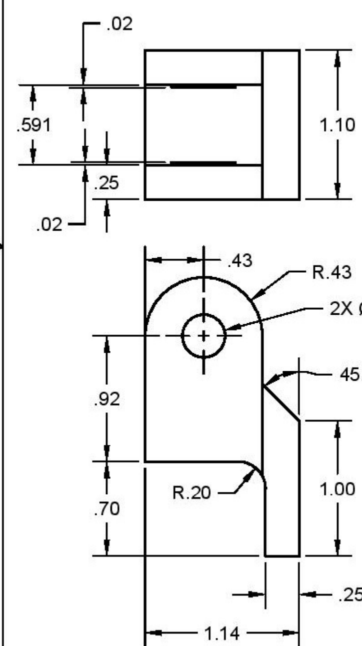
APPROVED	SIZE	CODE	DWG NO	TITLE	REV
CHECKED	A			Short Run Box Maker	
DRAWN Jesus Valdez 6/2/2020	SCALE 1:30		WEIGHT	SHEET 4/14	

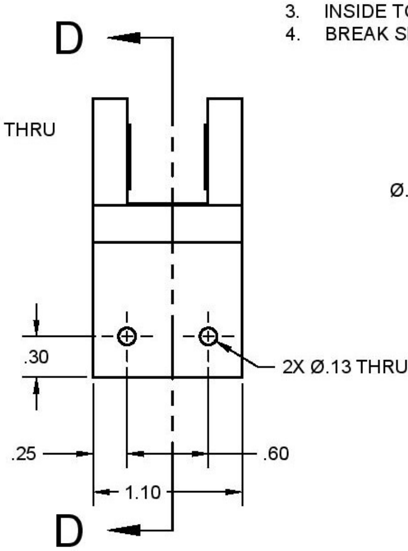


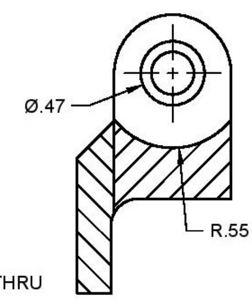
PART 9 - X LIMIT SWITCH MOUNT

NOTES:
MATERIAL: PLA PLASTIC
UNLESS OTHERWISE SPECIFIED:

1. ALL DIMS. IN INCHES
2. TOLERANCES
   X.XX = ±.01
   X.XXX = ±.005
   ANGLES = ±1°
3. INSIDE TOOL RADIUS .01 MAX.
4. BREAK SHARP EDGES .01 MAX.





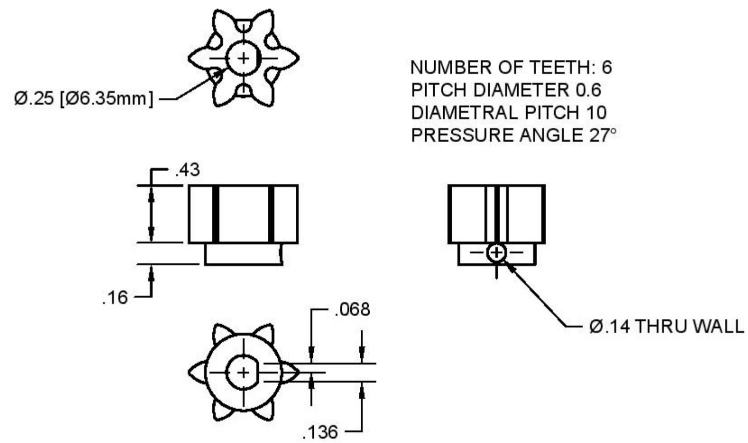


SECTION D-D
SCALE 1:1

PART 10 - EXIT BEARING MOUNT

APPROVED	SIZE	CODE	DWG NO	TITLE Short Run Box Maker	REV
CHECKED	A				
DRAWN Jesus Valdez 6/2/2020	SCALE 1:10		WEIGHT	SHEET 5/14	

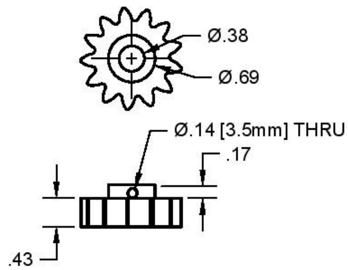
NUMBER OF TEETH: 6
PITCH DIAMETER 0.6
DIAMETRAL PITCH 10
PRESSURE ANGLE 27°



PART 11 - 6 TEETH SPUR GEAR

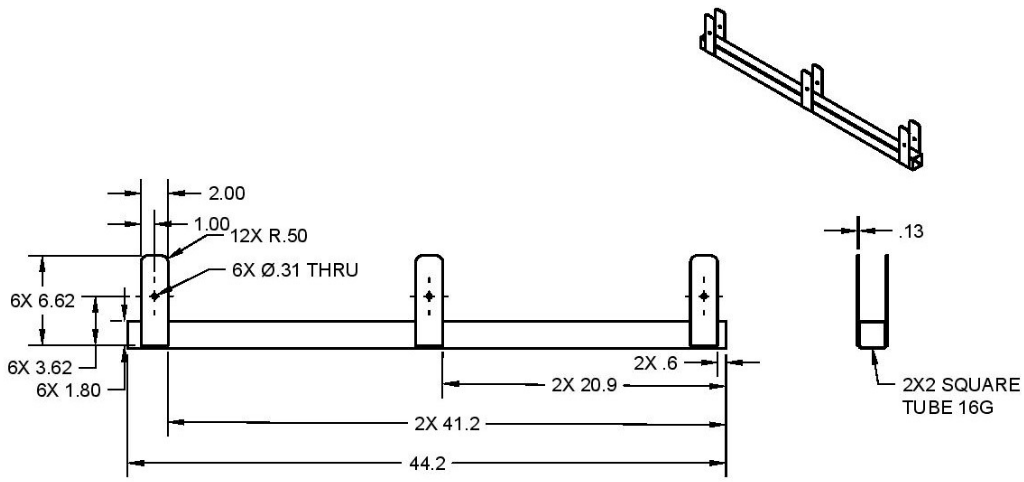
NOTES:
MATERIAL: PLA PLASTIC
UNLESS OTHERWISE SPECIFIED:
1. ALL DIMS. IN INCHES
2. TOLERANCES
   X.XX = ±.01
   X.XXX = ±.005
   ANGLES = ±1°
3. INSIDE TOOL RADIUS .01 MAX.
4. BREAK SHARP EDGES .01 MAX.

APPROVED	SIZE	CODE	DWG NO	TITLE	REV
CHECKED	A			Short Run Box Maker	
DRAWN Jesus Valdez 6/2/2020	SCALE 1:30		WEIGHT	SHEET 6/14	



NUMBER OF TEETH: 13
PITCH DIAMETER 1.3
DIAMETRAL PITCH 10
PRESSURE ANGLE 27°

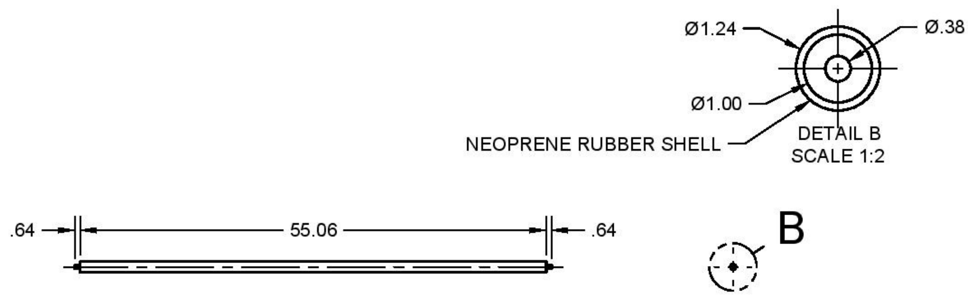
PART 12 - 13 TEETH SPUR GEAR



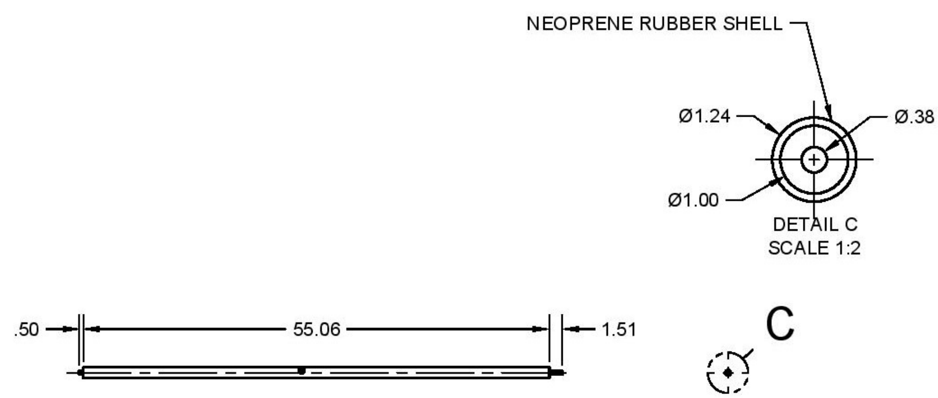
PART 13 - CARDBOARD GUIDE ASSEMBLY

NOTES:
MATERIAL:
PART 12: PLA PLASTIC
PART 13: 14G HOT ROLLED STEEL
UNLESS OTHERWISE SPECIFIED:
1. ALL DIMS. IN INCHES
2. TOLERANCES
   X.XX = ±.01
   X.XXX = ±.005
   ANGLES = ±1°
3. INSIDE TOOL RADIUS .01 MAX.
4. BREAK SHARP EDGES .01 MAX.

APPROVED	SIZE	CODE	DWG NO	TITLE	REV
CHECKED	A			Short Run Box Maker	
DRAWN Jesus Valdez 6/2/2020	SCALE 1:30		WEIGHT	SHEET 7/14	



PART 14 - FRONT ROLLER 1

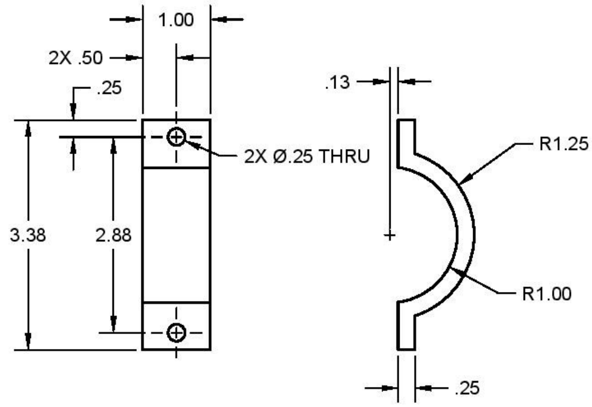


PART 15 - FRONT ROLLER 2

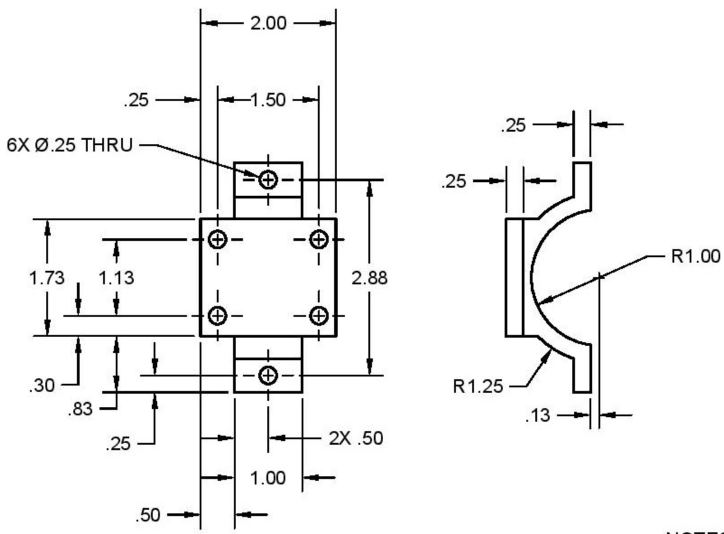
NOTES:
MATERIAL: 14G HOT ROLLED STEEL
UNLESS OTHERWISE SPECIFIED:

1. ALL DIMS. IN INCHES
2. TOLERANCES
   - X.XX = ±.01
   - X.XXX = ±.005
   - ANGLES = ±1°
3. INSIDE TOOL RADIUS .01 MAX.
4. BREAK SHARP EDGES .01 MAX.

APPROVED	SIZE	CODE	DWG NO	TITLE	REV
CHECKED	A			Short Run Box Maker	
DRAWN Jesus Valdez 6/2/2020	SCALE 1:30		WEIGHT	SHEET 8/14	



PART 16 - LASER TUBE CLAMP 1

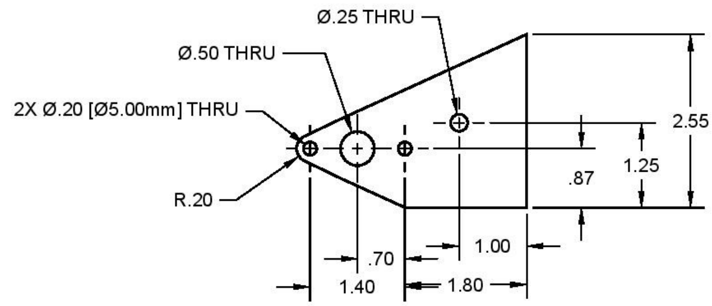


PART 17 - LASER TUBE CLAMP 2

NOTES:
MATERIAL: PLA PLASTIC
UNLESS OTHERWISE SPECIFIED:

1. ALL DIMS. IN INCHES
2. TOLERANCES
   - X.XX = ±.01
   - X.XXX = ±.005
   - ANGLES = ±1°
3. INSIDE TOOL RADIUS .01 MAX.
4. BREAK SHARP EDGES .01 MAX.

APPROVED		SIZE	CODE	DWG NO	TITLE	REV
CHECKED		A			Short Run Box Maker	
DRAWN Jesus Valdez	6/2/2020	SCALE 1:30		WEIGHT	SHEET 9/14	

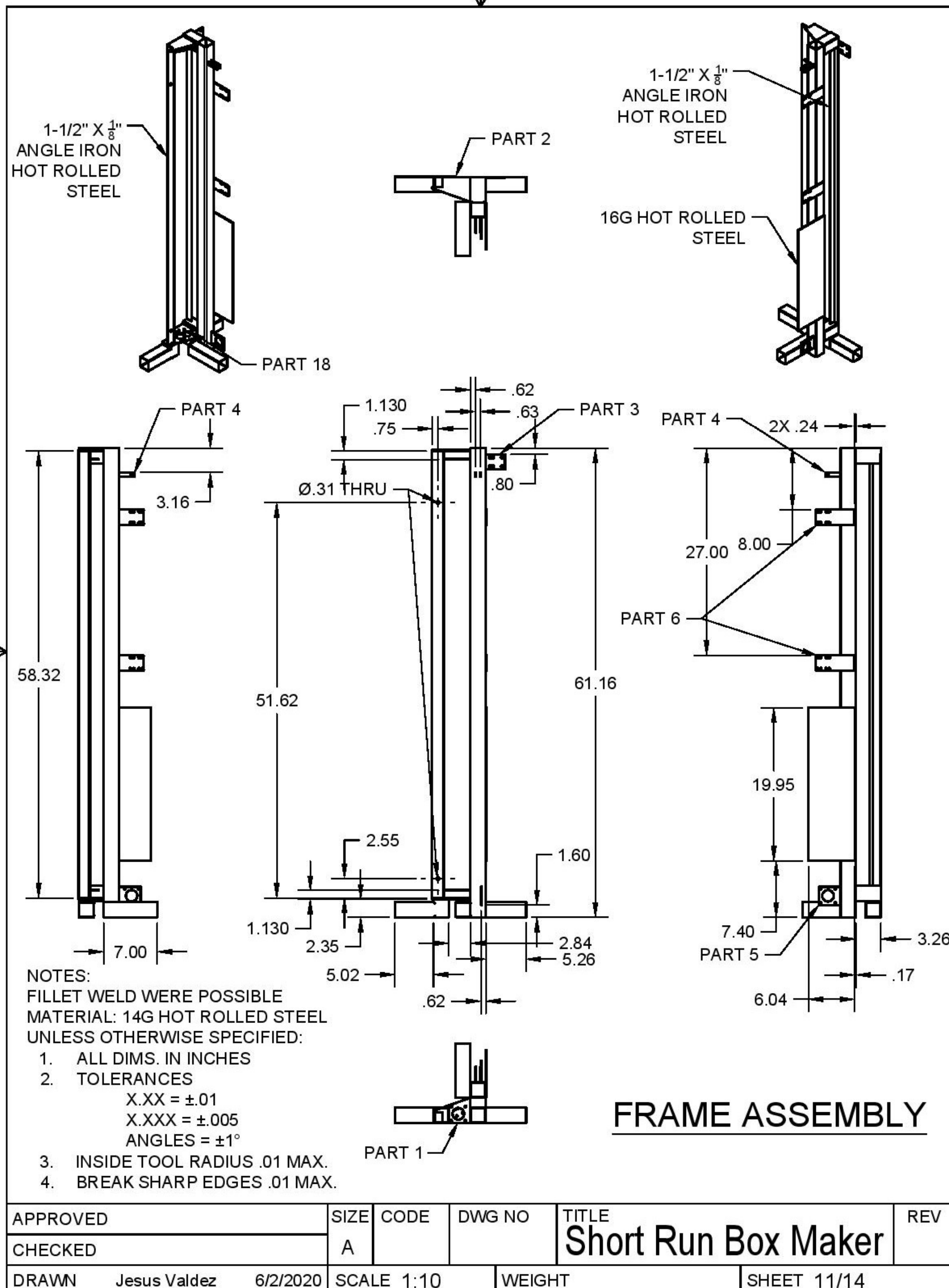


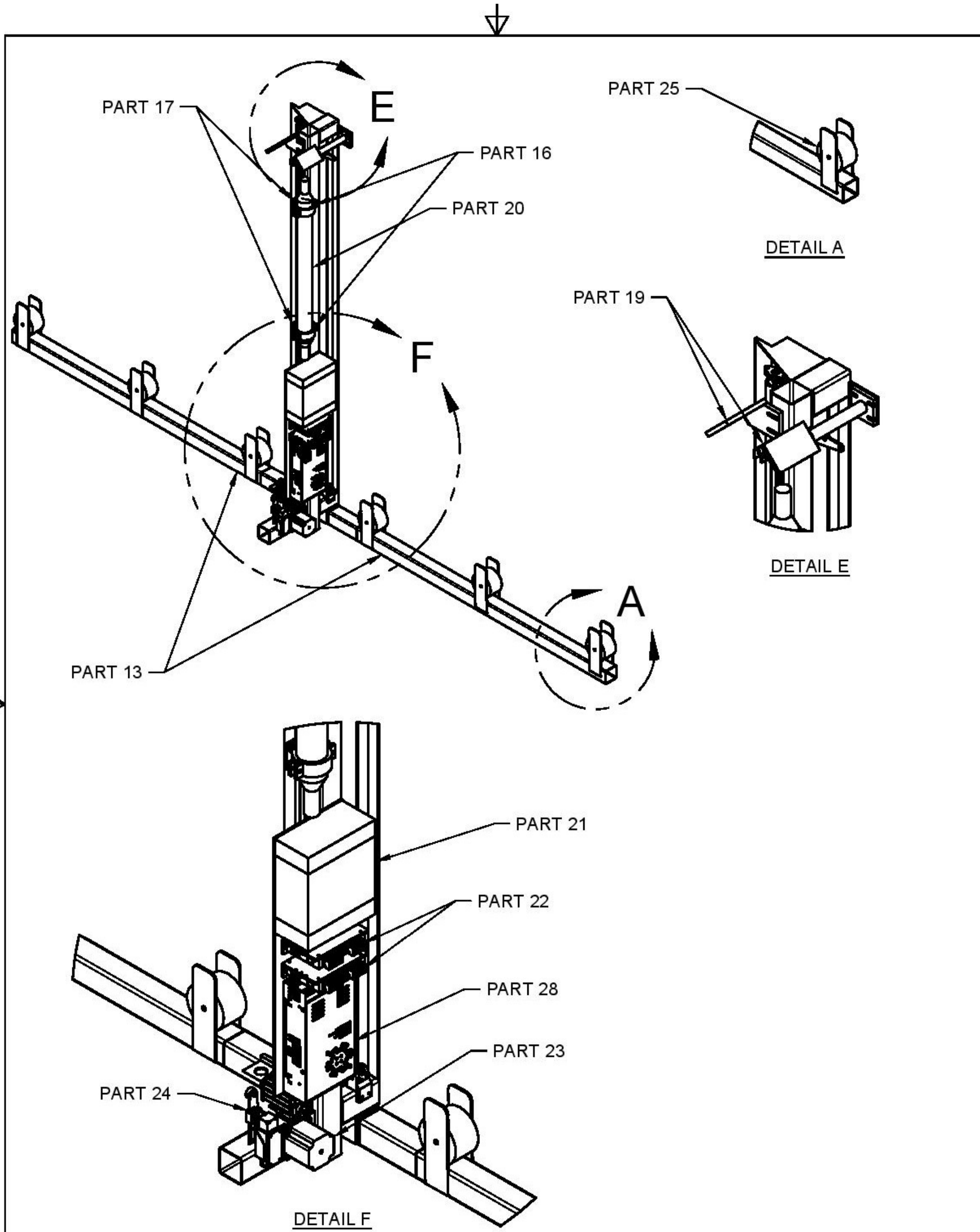
PART 18 - BOTTOM ROLLER BRACKET

NOTES:
MATERIAL: 14G HOT ROLLED STEEL
UNLESS OTHERWISE SPECIFIED:

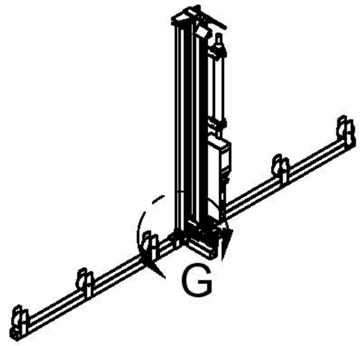
1. ALL DIMS. IN INCHES
2. TOLERANCES
   X.XX = ±.01
   X.XXX = ±.005
   ANGLES = ±1°
3. INSIDE TOOL RADIUS .01 MAX.
4. BREAK SHARP EDGES .01 MAX.

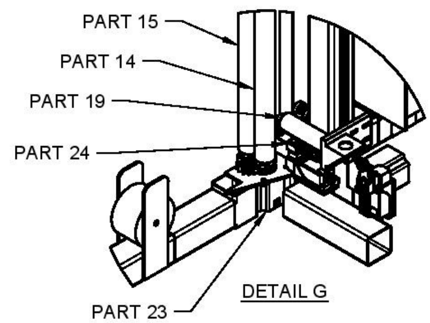
APPROVED	SIZE	CODE	DWG NO	TITLE Short Run Box Maker	REV
CHECKED	A				
DRAWN Jesus Valdez 6/2/2020	SCALE 1:30		WEIGHT	SHEET 10/14	

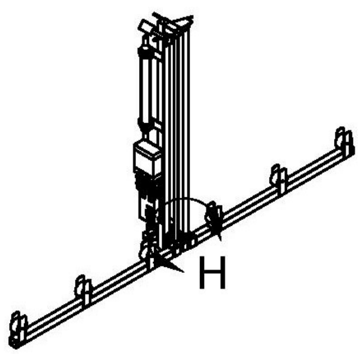

1-1/2" X 1/8" ANGLE IRON HOT ROLLED STEEL
PART 2
1-1/2" X 1/8" ANGLE IRON HOT ROLLED STEEL
16G HOT ROLLED STEEL
PART 18
PART 4
1.130
.75
.62
.63
PART 3
PART 4
2X .24
3.16
Ø.31 THRU
.80
27.00
8.00
PART 6
58.32
51.62
61.16
19.95
2.55
1.60
1.130
2.35
2.84
5.26
7.40
PART 5
3.26
7.00
5.02
.62
.17
6.04
NOTES:
FILLET WELD WERE POSSIBLE
MATERIAL: 14G HOT ROLLED STEEL
UNLESS OTHERWISE SPECIFIED:
1. ALL DIMS. IN INCHES
2. TOLERANCES
X.XX = ±.01
X.XXX = ±.005
ANGLES = ±1°
3. INSIDE TOOL RADIUS .01 MAX.
4. BREAK SHARP EDGES .01 MAX.
PART 1
FRAME ASSEMBLY
APPROVED
CHECKED
SIZE A
CODE
DWG NO
TITLE Short Run Box Maker
REV
DRAWN Jesus Valdez 6/2/2020
SCALE 1:10
WEIGHT
SHEET 11/14

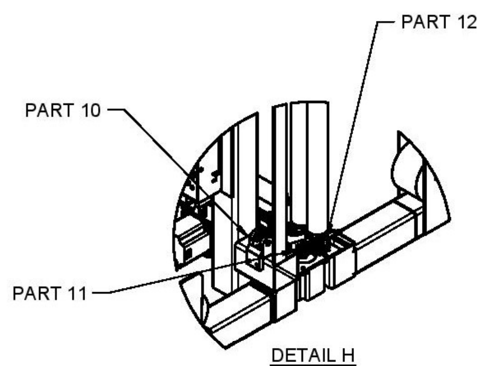


APPROVED	SIZE A	CODE	DWG NO	TITLE Short Run Box Maker	REV
CHECKED					
DRAWN Jesus Valdez 6/2/2020	SCALE 1:10		WEIGHT	SHEET 12/14	









APPROVED	SIZE	CODE	DWG NO	TITLE Short Run Box Maker	REV
CHECKED	A				
DRAWN Jesus Valdez 6/2/2020	SCALE 1:30		WEIGHT	SHEET 13/14	

10. Bill of Materials

Part No.	Name	Qty.
1	Bottom Mount	1
2	Top Mount	1
3	Mirror Mount	1
4	X Axis Pulley Mount	2
5	X Motor Mount	1
6	Laser Frame Bracket	2
7	Top Roller Bracket	2
8	Y Limit Switch Mount	1
9	X Limit Switch Mount	1
10	Exit Bearing Mount	1
11	6 Teeth Spur Gear	1
12	13 Teeth Spur Gear	1
13	Cardboard Guide Assembly	2
14	Front Roller 1	1
15	Front Roller 2	1
16	Laser Tube Clamp 1	2
17	Laser Tube Clamp 2	2
18	Bottom Roller Bracket	1
19	Cloudray C Series Co2 Laser Head Set	1
20	Cloudray 40 W Co2 Glass Laser Tube	1
21	Cloudray 40 W Co2 Laser Power Supply	1
22	TB6600 Stepper Motor Driver	2
23	STEPPERONLINE Nema 23	2
24	HiLetgo Momentary Limit Switch	2
25	58mm Skateboard Wheels	6
26	608-2RS Bearing	6
27	Uxcell KFL08 Pillow Block	4
28	MENZO 12V Power Supply	1
29	M8-1.25 Nylon Hex nut	9
30	M8-1.25 65mm Socket Head Cap Screw	6
31	M5-0.8 Hex Nut	22
32	M5-0.8 12mm Socket Head Cap Screw	12
33	M5-0.8 15mm Socket Head Cap Screw	8
34	M4-0.7 30mm Socket Head Cap Screw	4
35	M4-0.7 18mm Socket Head Cap Screw	4
36	M4-0.7 Hex Nut	8
37	M4-0.7 10mm Socket Head Cap Screw	4
38	1/4-20 1" Socket Head Cap Screw	4

39	1/4-20 Hex Nut	16
40	#14x1" Self Tapping Hex Head Screw	4
41	1/4-20 .5" Socket Head Cap Screw	2
42	GT2 Idler Pulley	1
43	GT2 20T Pulley	1
44	M8-1.25 25mm Socket Head Cap Screw	2
45	M8-1.25 35mm Socket Head Cap Screw	1