MULTIDISCIPLINARY SENIOR DESIGN PROJECT
GE 498
COLLEGE OF ENGINEERING
VALPARAISO UNIVERSITY
VALPARAISO, INDIANA

Design Documentation
for
Biaxial Tensile Tester

TEAM TENSILE TESTER
Date: May 5th, 2022

Prepared by:	**Team Tensile Tester**	
Concurrence:	G. El-Howayek	Initials:
Approved (Customer):	Bethany Luke	Initials:
Approved (Team Leader):	Michael Scarsella	Initials: MS

Finalized Design Requirements:

Goal Statement

The Biaxial Tensile Tester is a device that will test and record the strength of skin, organs and other organic tissues while pulling on the tissue simultaneously in two orthogonal directions.

Objectives

In order to achieve success, the system shall

1. Be able to pull tissue with the same velocity from all 4 sides simultaneously
2. Be able to measure and record the force and displacement
3. Be able to be set up easily and operated safely
4. Be lightweight and portable

System Requirements

1. TEST REQUIREMENTS

    1.1 The tissue sample shall be suspended in the air or a fluid during the test to avoid friction on a surface that would distort the measurements.
    1.2 The system shall apply 0.1 Newtons (in tension) to remove slack from the specimen and threads (if using) before deforming the tissue
    1.3 The data shall be zeroed after the slack is removed.
    1.4 The system shall deform tissue in 2 orthogonal directions simultaneously, where the angle between the two directions is within 1 degree of 90.
    1.5 The tissue shall deform at a constant velocity of 50 +/- 0.1 mm/min in both the x axis and the y axis, which are two orthogonal directions in the horizontal plane.
    1.6 The system shall maintain an even distribution of forces across each edge of the tissue every 0.1 to 0.5 +/- 0.1 cm each side.
    1.7 The system shall be able to apply at least 80 Newtons (in tension) of force in both the x direction and the y direction.
    1.8 The initial size of the specimen shall be between 1 cm by 1 cm and 4 cm by 4 cm.
    1.9 The maximum positive displacement in both the x and y direction shall be at least 6 cm.

2. DATA REQUIREMENTS

    2.1 The system shall record the actual forces applied within +/- 0.1 Newtons in the x direction and in the y direction.
    2.2 The system shall record the displacement of the tissue within +/- 0.2 mm in the x direction and in the y direction.
    2.3 The data shall be exported to a spreadsheet on a computer, SD card, or flash drive.

3. SAFETY AND EASE OF OPERATION REQUIREMENTS

    3.1 [1] The electronics shall have an IPX-1 rating.
    3.2 The system shall have a start and stop button for collecting data.
    3.3 The system shall have an emergency stop function.
    3.4 An untrained student shall be able to learn how to set up the device and perform tests after 30-60 minutes of training.

4. PORTABILITY REQUIREMENTS

    4.1 The system footprint shall be smaller than 3x3ft
    4.2 The system shall not exceed 6 feet in the vertical direction.
    4.3 The system shall be powered by a 110V wall outlet.

Desired Criteria

    5.1 The velocity may be adjusted to 20 mm/min in the x and y axis.
    5.2 The velocity may be adjusted to 100 mm/min in the x and y axis.
    5.3 The velocity may be adjusted to a constant velocity entered by the user (mm/min) in the x and y axis.
    5.4 The system may include a fluid bath around the specimen at a temperature of 37 degrees C.
    5.5 The system may record displacement with a camera above, where the frames are synchronized with the force data.

Works Cited

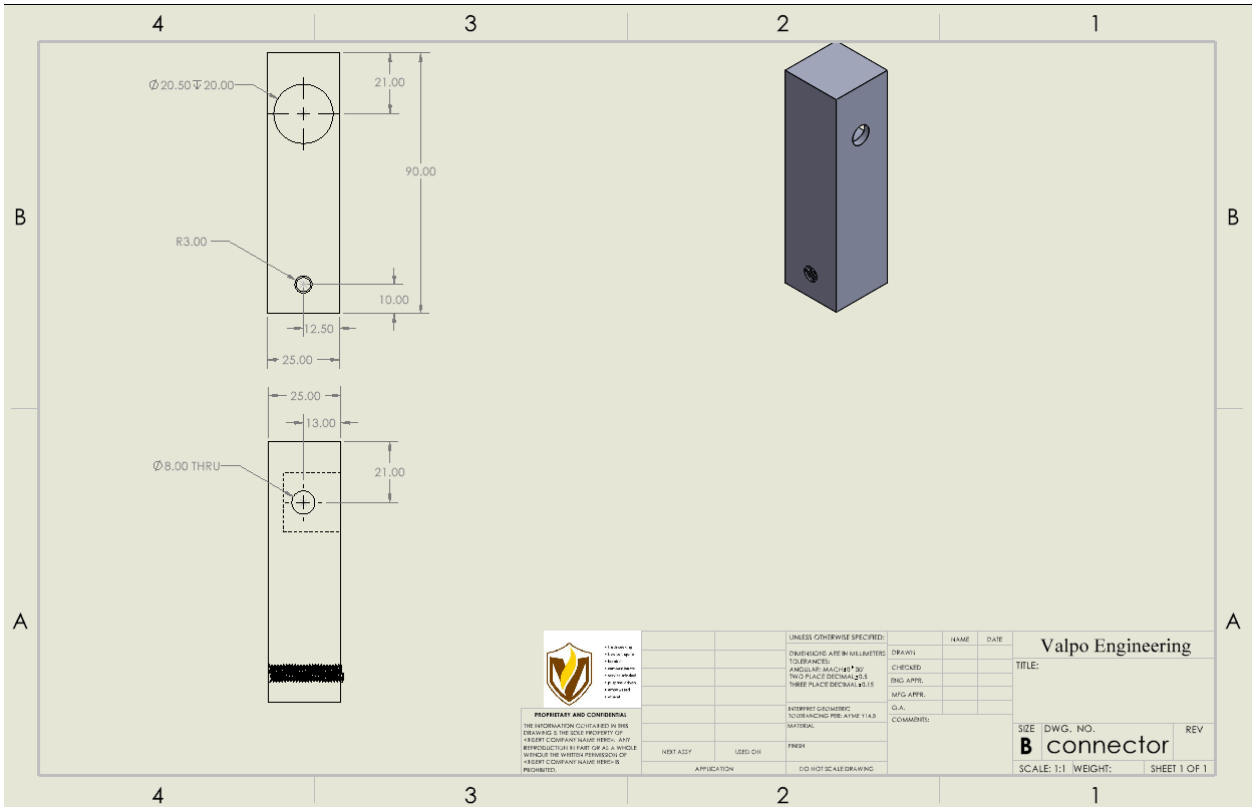
[1] Alam, Noor, 2020, “IPX WaterProof Rating” [Online] Available: https://speakersmag.com/ipx-waterproof-rating-ipx4-ipx5-ipx6-ipx7-ipx8-ratings [Accessed: 7-September-2021]

Parts list, including supplier, cost, quantity:

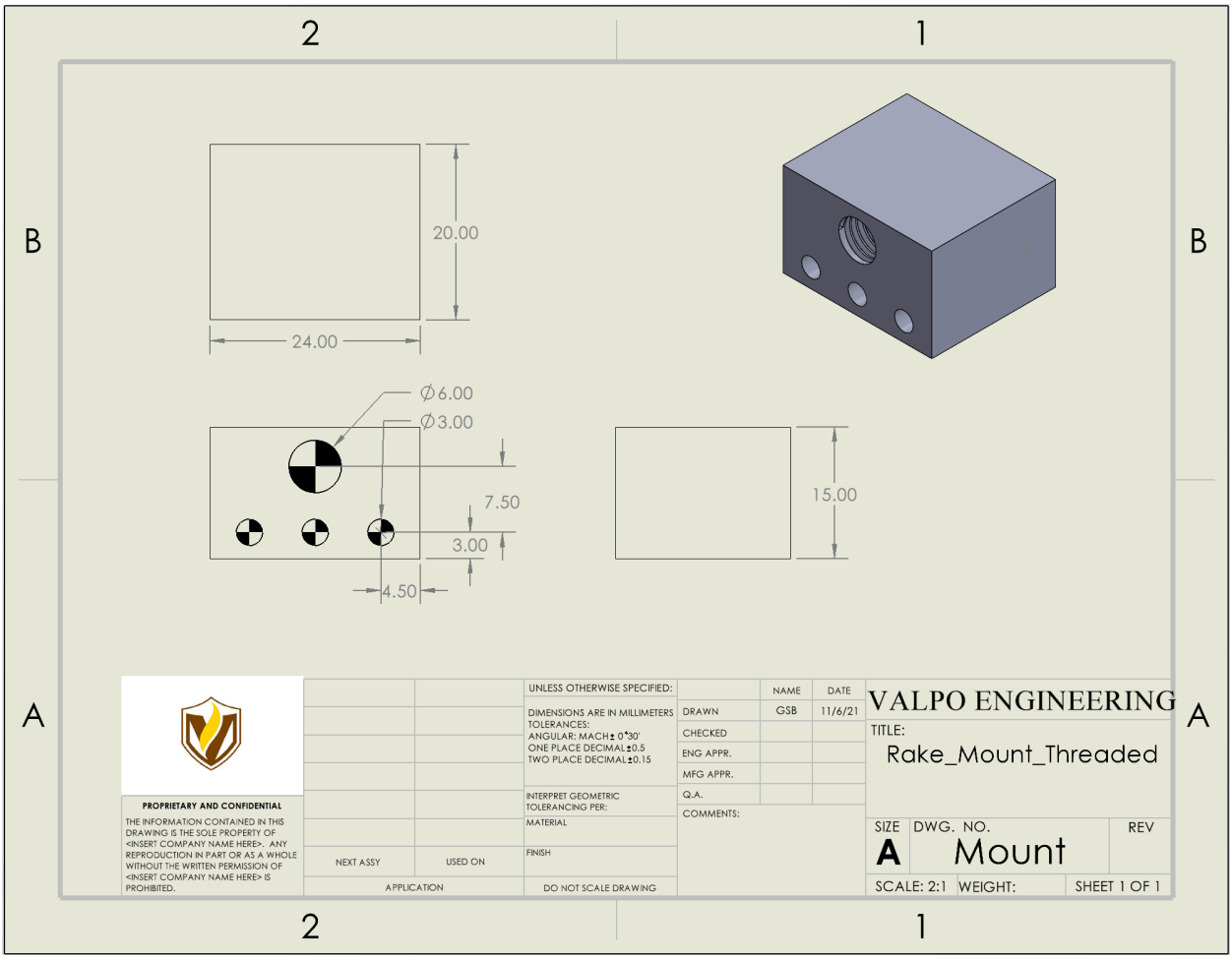
Part/Material	Quantity	Total Cost	Part Number/Code	Manufacturer	Supplier
4" Linear Actuator	4	$290.84	Z0600WH	Zoom Industrial	Amazon
Arduino	1	$43.07	A000067	Arduino	Arduino
Arduino Shield	1	$34.24	HD025	Xiken Electronic Technology Co., Ltd	Amazon
Arduino Cable	1	$6.50	M000006	Arduino	Arduino
Motor Driver (2 Pack)	2	$29.98	BTS7960	Songhe	Amazon
110VAC to 12VDC Converter	1	$27.99	B08ZJZPF4Y	LEDneighbor	Amazon
Rods	4	$25.64	a19102800ux0069	uxcell	Amazon
Eosa 1x100	3	$17.97	g20080400ux0007	uxcell	Amazon
Brackets	1	$19.25	SDWCB01	Alamic	Amazon
Base bars	1	$174	6880	IXGNIJ	Amazon
	1	$23.52	TL-2028-20B	Tongling Hardware	Amazon
Base connectors	1	$17.11	AM-XC5265	Boeray	Amazon
Blocks	1	$25.36	R3303=NK	WeatherShield	Home Depot
Load Cell	2	$147.95	Walfrontrzgkc28bxn-02	Walfront	Amazon
Load Cell Amplifier	2	$24.83	SEN-13879	SparkFun Electronics	Amazon
Junction Box	1	$30.00	FL-1859-330	SockitBoX	Amazon
Stock	1	$36.55	SQ31	MetalsDepot	MetalsDepot
Set Screws	6	$9.06	812168	Everbilt	Home Depot
6 Channel Encoder	1	$84.99		robogaia	robogaia
Encoder Shield Connector	4	$11.96		robogaia	robogaia
Screws	1	$7.97	114SDDW1	Grip-Rite	Home Depot
Total		$1,088.78			

Detailed drawings and assembly drawings:

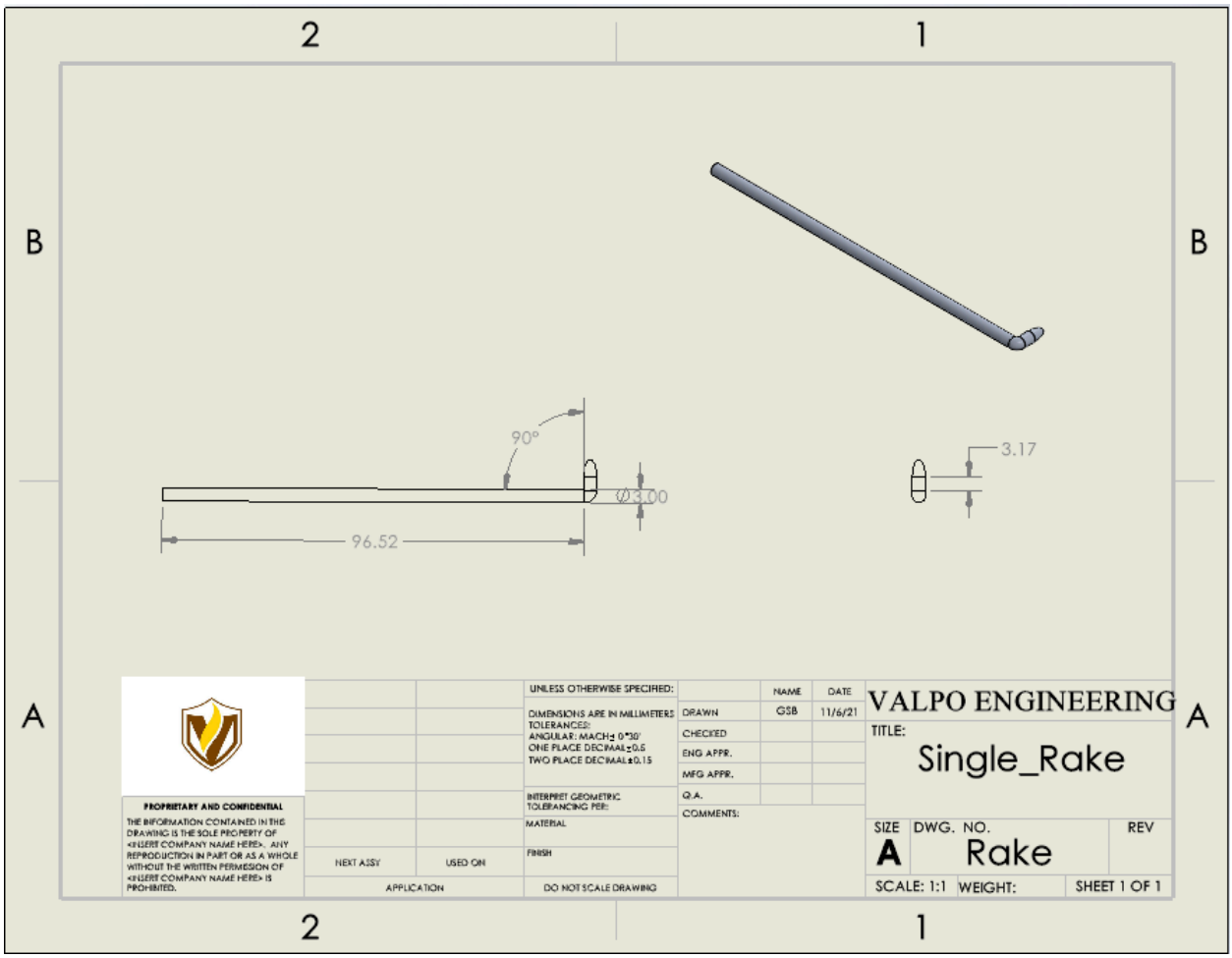
- Vertical connector bar:



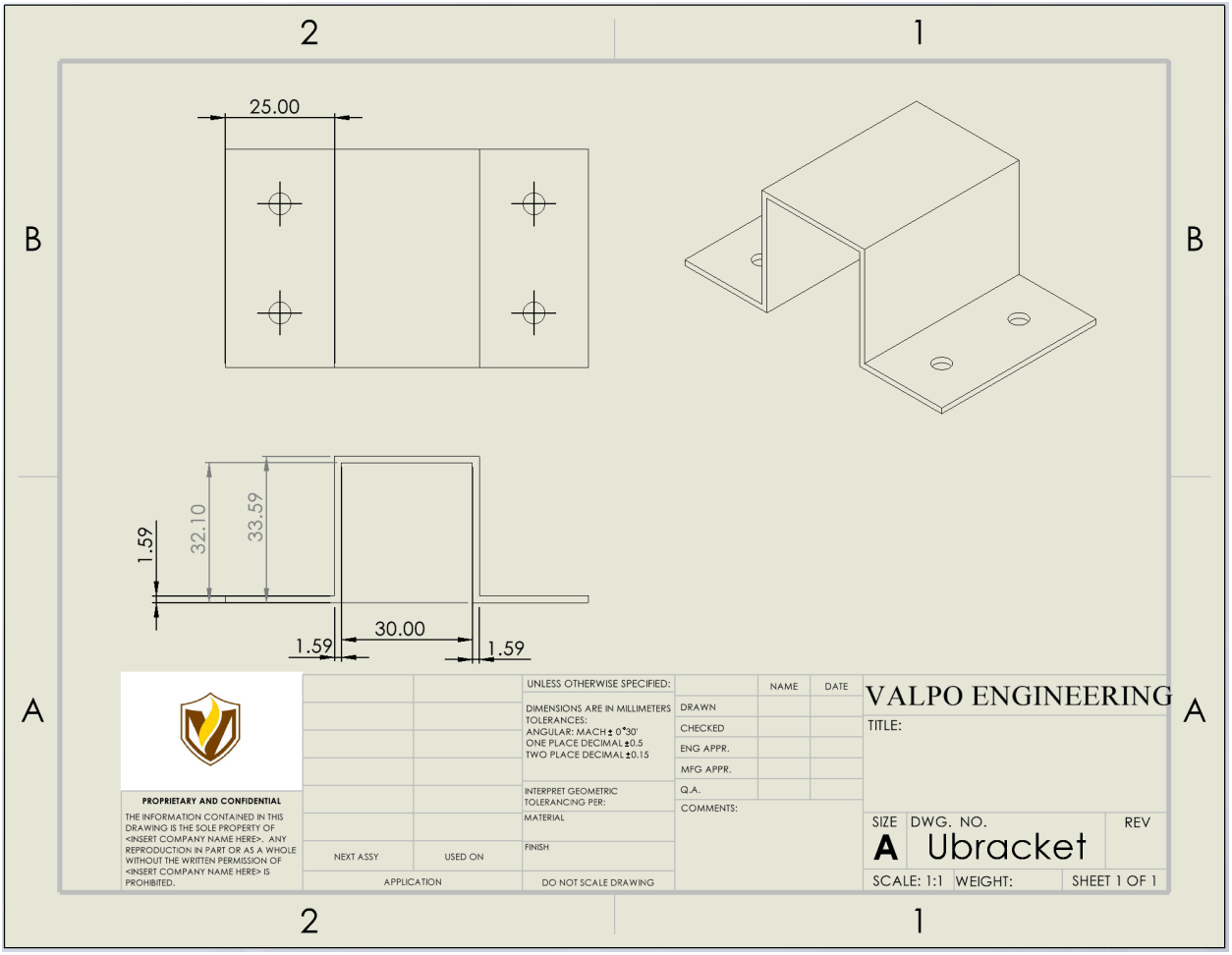
- 3mm rake block:



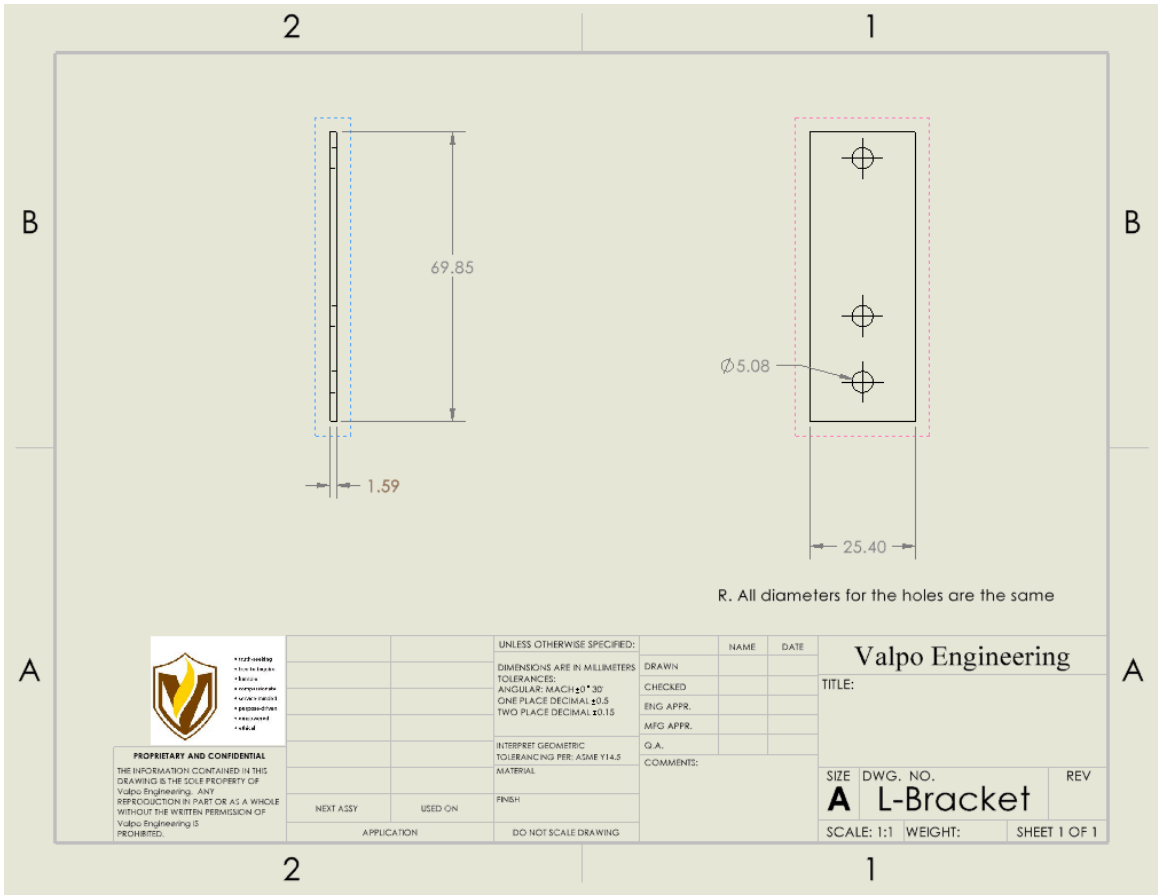
- Individual rake:



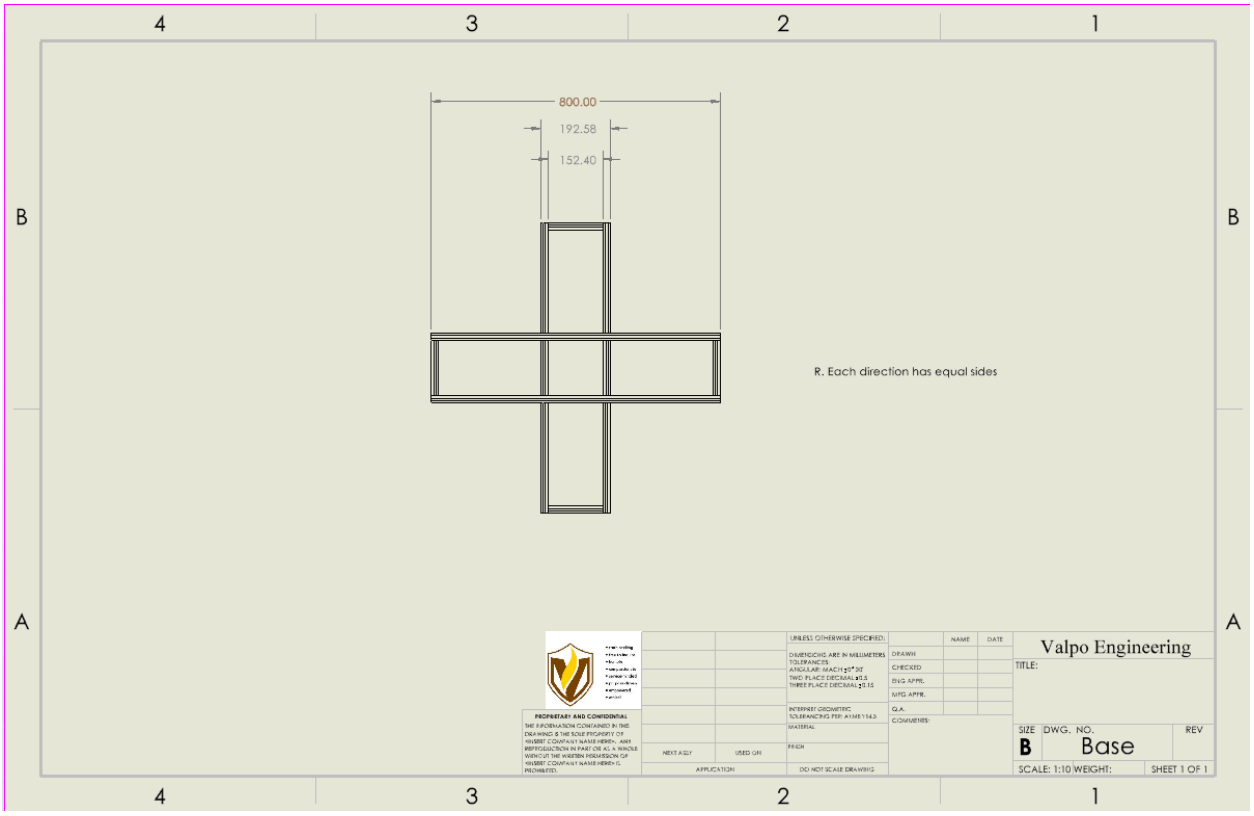
- U-Bracket


2
1
B
A
25.00
32.10
33.59
1.59
30.00
1.59
1.59
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
TOLERANCES:
ANGULAR: MACH ± 0°30'
ONE PLACE DECIMAL ±0.5
TWO PLACE DECIMAL ±0.15
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
DO NOT SCALE DRAWING
NEXT ASSY
USED ON
APPLICATION
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
VALPO ENGINEERING
TITLE:
SIZE
A
DWG. NO.
Ubracket
REV
SCALE: 1:1
WEIGHT:
SHEET 1 OF 1

- L-Bracket

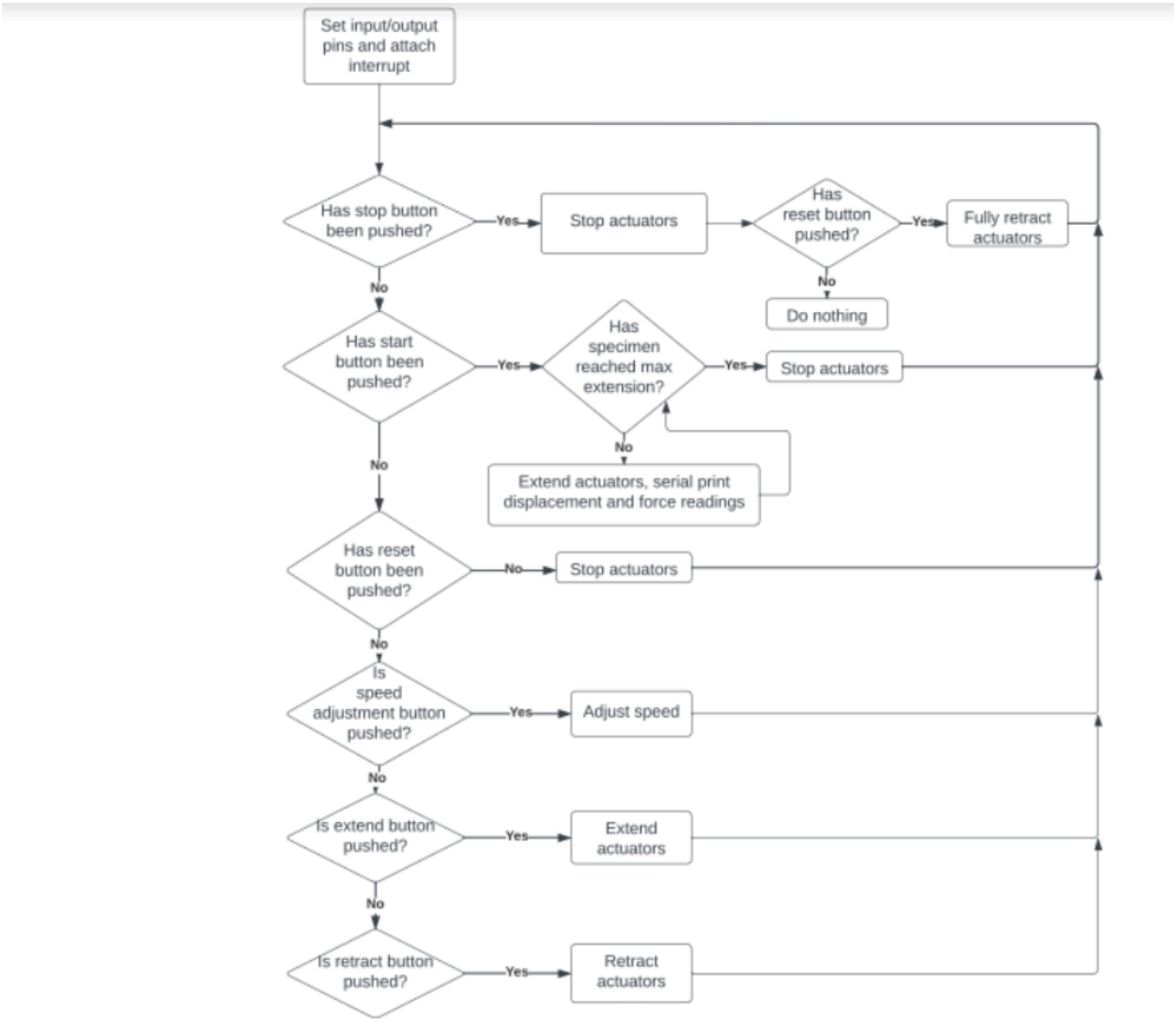

2
1
B
A
69.85
1.59
Ø5.08
25.40
R. All diameters for the holes are the same
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
TOLERANCES:
ANGULAR: MACH ±0° 30'
ONE PLACE DECIMAL ±0.5
TWO PLACE DECIMAL ±0.15
INTERPRET GEOMETRIC TOLERANCING PER: ASME Y14.5
MATERIAL
FINISH
DO NOT SCALE DRAWING
NEXT ASSY
USED ON
APPLICATION
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
Valpo Engineering
TITLE:
SIZE
A
DWG. NO.
L-Bracket
REV
SCALE: 1:1
WEIGHT:
SHEET 1 OF 1
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF Valpo Engineering. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF Valpo Engineering IS PROHIBITED.

- Base (plus shape)

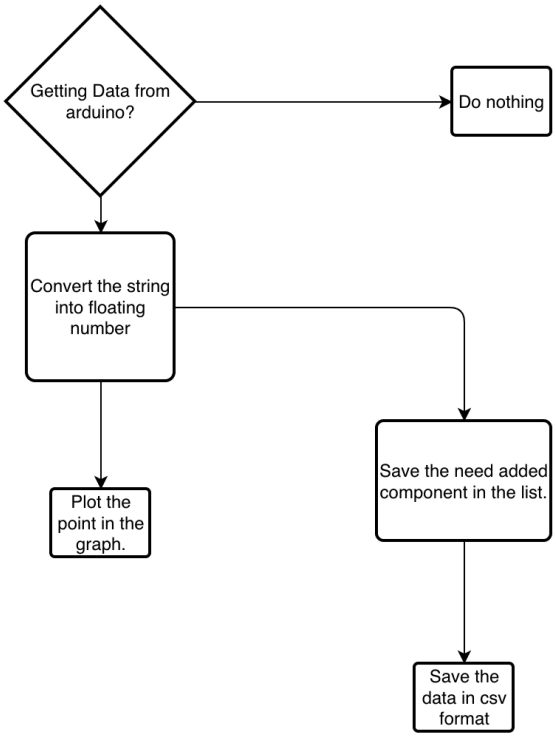


Control algorithm flow charts:

- Arduino flowchart



- Python flow chart



Compute code:

- Arduino code

```
#include "HX711.h"

#define DOUT  4
#define CLK  3
#define DOUT2  25
#define CLK2  24

HX711 scale;
HX711 scale2;

float calibration_factor = -51000; //roughly -51000 works for 10kg load cell at 0.5 to 0.7kg

bool emergencyStop = false;
bool start = false;
long pos = 0;
long prevPos = 1;
long steps = 0;
long prevSteps = 0;
float conNum = 0.000286 * 3.045 * 8;
bool dir = 0; //1 = retract, 0 = extend
int Speed1 = 245;
int Speed2 = 245;
int Speed3 = 245;
int Speed4 = 245;
int vinc = 5;
int vincSmall = 1;
int vincLarge = 10;
bool homeFlag;
unsigned long prevTimer = 0;
unsigned long lastStepTime = 0;
int trigDelay = 500;
int count = 0;
```

```
float force1 = 0.0;
float force2 = 0.0;

void updatePosition(void) {
  if (dir == 1) {
    pos = pos + steps;
    steps = 0;
  } else {
    pos = pos - steps;
    steps = 0;
  }
}


float convertToInches(long pos) {
  return conNum * pos;
}

void setup() {
  /*Serial.begin(9600);
    Serial.println("HX711 calibration sketch");
    Serial.println("Remove all weight from scale");
    Serial.println("After readings begin, place known weight on scale");
    Serial.println("Press + or a to increase calibration factor");
    Serial.println("Press - or z to decrease calibration factor");*/

  scale.begin(DOUT, CLK);
  scale.set_scale();
  scale.tare(); //Reset the scale to 0

  scale2.begin(DOUT2, CLK2);
  scale2.set_scale();
  scale2.tare(); //Reset the scale to 0

  long zero_factor = scale.read_average(); //Get a baseline reading
  long zero_factor2 = scale2.read_average(); //Get a baseline reading
```

```
  /*Serial.print("Zero factor: "); //This can be used to remove the need to tare the scale. Useful in permanent scale projects.
    Serial.println(zero_factor);*/

  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);

  pinMode(22, INPUT_PULLUP); //start button
  pinMode(23, INPUT_PULLUP); //reset button
  pinMode(35, INPUT_PULLUP); //emergency stop button
  pinMode(36, INPUT_PULLUP); //velocity increment 10 button
  pinMode(37, INPUT_PULLUP); //velocity decrement 10 button
  pinMode(48, INPUT_PULLUP); //vinc 1 increase button
  pinMode(51, INPUT_PULLUP); //vinc 1 decrease button
  pinMode(38, INPUT_PULLUP); //displacement adjustment button 1
  pinMode(39, INPUT_PULLUP); //displacement adjustment button 1

  pinMode(2, INPUT);
  attachInterrupt(digitalPinToInterrupt(2), countSteps, RISING);

  Serial.begin(9600);
}

void printSpeeds() {
  Serial.print("Speed1: ");
  Serial.print(Speed1);
  Serial.print(" Speed2: ");
  Serial.print(Speed2);
  Serial.print(" Speed3: ");
  Serial.print(Speed3);
  Serial.print(" Speed4: ");
```

```
  Serial.println(Speed4);
}

void extend() {
  dir = 0;
  analogWrite(6, 0);
  analogWrite(7, Speed1);
  analogWrite(8, 0);
  analogWrite(9, Speed2);
  analogWrite(10, 0);
  analogWrite(11, Speed3);
  analogWrite(12, 0);
  analogWrite(13, Speed4);
}

void retract() {
  dir = 1;
  analogWrite(6, Speed1);
  analogWrite(7, 0);
  analogWrite(8, Speed2);
  analogWrite(9, 0);
  analogWrite(10, Speed3);
  analogWrite(11, 0);
  analogWrite(12, Speed4);
  analogWrite(13, 0);
}

void stopActuators() {
  analogWrite(6, 0);
  analogWrite(7, 0);
  analogWrite(8, 0);
  analogWrite(9, 0);
  analogWrite(10, 0);
  analogWrite(11, 0);
  analogWrite(12, 0);
  analogWrite(13, 0);
}
```

```
void printButtons(){
  Serial.print(digitalRead(51));
  Serial.print("    ");
  Serial.print(digitalRead(48));
  Serial.print("    ");
  Serial.print(digitalRead(37));
  Serial.print("    ");
  Serial.print(digitalRead(36));
  Serial.print("    ");
  Serial.print(digitalRead(38));
  Serial.print("    ");
  Serial.print(digitalRead(39));
  Serial.print("    ");
  Serial.print(digitalRead(22));
  Serial.print("    ");
  Serial.print(digitalRead(35));
  Serial.print("    ");
  Serial.print(digitalRead(23));
}

void loop() {

  if (emergencyStop == false) {

    if (digitalRead(35) == LOW) { //emergency stop button is pressed
      emergencyStop = true;
    }

    if (start == false) {

      force1 = scale.get_units() * 0.453592 * 9.81, 2;
      if (force1 < 0.0) {
        force1 = 0.0;
      }
      force2 = scale2.get_units() * 0.453592 * 9.81, 2;
      if (force2 < 0.0) {
```

```
  force2 = 0.0;
 }
 Serial.print((String)force1);
 Serial.print(",");
 Serial.print((String)convertToInches(pos));
 Serial.print(",");
 Serial.print((String)force2);
 Serial.print(",");
 Serial.print((String)convertToInches(pos));
 Serial.print(",");
 Serial.print((String)Speed1);
 Serial.println("");
 //delay(250);

 if (digitalRead(22) == LOW) { //If button is pressed once
  start = true; //Start test
 }

 if (digitalRead(23) == LOW) { //If reset button is pressed
  start = false;
  homeFlag = 0;
  homeActuator();
 }

 if (digitalRead(37) == LOW & Speed1 >= 1 + vincLarge & Speed2 >= 1+
vincLarge & Speed3 >= 1 + vincLarge & Speed4 >= 1 + vincLarge) { //velocity
increment is pressed
  delay(250);
  Speed1 -= vincLarge;
  Speed2 -= vincLarge;
  Speed3 -= vincLarge;
  Speed4 -= vincLarge;
  //printSpeeds();

 }
```

```
    else if (digitalRead(36) == LOW & Speed1 <= 255 - vincLarge & Speed2 <=
255 - vincLarge & Speed3 <= 255 - vincLarge & Speed4 <= 255 - vincLarge) {
//velocity decrement is pressed
      delay(250);
      Speed1 += vincLarge;
      Speed2 += vincLarge;
      Speed3 += vincLarge;
      Speed4 += vincLarge;
      //printSpeeds();
    }

    if (digitalRead(51) == LOW & Speed1 >= vincSmall + 1 & Speed2 >=
vincSmall + 1 & Speed3 >= vincSmall + 1 & Speed4 >= vincSmall + 1) {
//increase vinc button is pressed
      delay(250);
      Speed1 -= vincSmall;
      Speed2 -= vincSmall;
      Speed3 -= vincSmall;
      Speed4 -= vincSmall;
      //printSpeeds();
    }

    else if (digitalRead(48) == LOW & Speed1 <= 255 - vincSmall & Speed2 <=
255 - vincSmall & Speed3 <= 255 - vincSmall & Speed4 <= 255 - vincSmall) {
//decrease vinc button is pressed
      delay(250);
      Speed1 += vincSmall;
      Speed2 += vincSmall;
      Speed3 += vincSmall;
      Speed4 += vincSmall;
      //printSpeeds();
    }


    if (homeFlag == 0) {
      homeActuator();
    }
```

```
    //Serial.println(digitalRead(8));
    //Serial.println(digitalRead(9));

    if (digitalRead(38) == HIGH & digitalRead(39) == LOW) {
      //Extend actuator
      extend();
      if (millis() - prevTimer > 100) {
        updatePosition();
        prevTimer = millis();
        if (pos == prevPos | pos == 0) {
          pos = 0;
        }
        else {
          prevPos = pos;
        }
        /*Serial.print("Displacement: ");
          Serial.print(convertToInches(pos));
          Serial.print(" inches         ");

          //Only read force immediately after displacement
          Serial.print("Force: ");
          Serial.print(scale.get_units() * 0.453592 * 9.81, 2); //*9.81 = kg to Newton
conversion
          Serial.println(" Newtons");*/
      }
    }
    else if (digitalRead(38) == LOW & digitalRead(39) == HIGH) {
      //Retract Actuator
      retract();
      if (millis() - prevTimer > 100) {
        updatePosition();
        prevTimer = millis();
        if (pos == prevPos | pos == 574) {
          pos = 574;
        }
        else {
```

```
        prevPos = pos;
       }
       /* Serial.print("Displacement: ");
        Serial.print(convertToInches(pos));
        Serial.print(" inches         ");

        //Only read force immediately after displacement
        Serial.print("Force: ");
        Serial.print(scale.get_units() * 0.453592 * 9.81, 2); //*9.81 = kg to Newton
conversion
        Serial.println(" Newtons");*/
      }
     }
     else {
      stopActuators();
      if (digitalRead(22) == LOW) { //If button is pressed once
       start = true; //Start test
      }
     }

     scale.set_scale(calibration_factor); //Adjust to this calibration factor
     scale2.set_scale(calibration_factor);

     if (Serial.available())
     {
      char temp = Serial.read();
      if (temp == '+' || temp == 'a')
       calibration_factor += 100;
      else if (temp == '-' || temp == 'z')
       calibration_factor -= 100;
     }
    }
    else { //Running test

     //2 is a PLACEHOLDER VALUE for length of test
     if (convertToInches(pos) >= 2) {
      start = false;
```

```
      //Serial.println("");
      //Serial.println(count);
    }
    else {
      //Retract actuator
      retract();
      if (millis() - prevTimer > 100) {
        updatePosition();
        prevTimer = millis();
        if (pos == prevPos | pos == 574) {
          pos = 574;
        }
        else {
          prevPos = pos;
        }
        /* Serial.print("Displacement: ");
           Serial.print(convertToInches(pos));
           Serial.print(" inches         ");

           //Only read force immediately after displacement
           Serial.print("Force: ");
           Serial.print(scale.get_units() * 0.453592 * 9.81, 2); //*9.81 = kg to
Newton conversion
           Serial.println(" Newtons");*/

        force1 = scale.get_units() * 0.453592 * 9.81, 2;
        if (force1 < 0.0) {
          force1 = 0.0;
        }
        force2 = scale2.get_units() * 0.453592 * 9.81, 2;
        if (force2 < 0.0) {
          force2 = 0.0;
        }
        //Output for Python graphing
        Serial.print((String)force1);
        Serial.print(",");
        Serial.print((String)convertToInches(pos));
```

```
        Serial.print(",");
        Serial.print((String)force2);
        Serial.print(",");
        Serial.print((String)convertToInches(pos));
        Serial.print(",");
        Serial.print((String)Speed1);
        Serial.println("");
        delay(250);

        count++;
       }
      }
    }
  }
  else {

    if (digitalRead(23) == LOW) { //If reset button is pressed

      homeFlag = 0;
      homeActuator();
      start = false;
      emergencyStop = false;
    }

    //Stop
    stopActuators();

  }
}

void countSteps(void) {
  if (micros() - lastStepTime > trigDelay) {
    steps++;
    lastStepTime = micros();
  }
}
```

```
void homeActuator(void) { //retract all the way
  prevTimer = millis();
  while (homeFlag == 0) {
    retract();
    if (prevSteps == steps) {
      if (millis() - prevTimer > 100) {
        stopActuators();
        steps = 0;
        homeFlag = 1;
        pos = 574;
      }
    } else {
      prevSteps = steps;
      prevTimer = millis();
    }
  }
}
```

- Python code for the interface

```
import serial
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.ticker import MultipleLocator
from datetime import datetime

filename="data.csv"

file=open(filename,"a")
file.write("Time,Force1,Displacement1,Force2,Displacement2,Speed\n")
file.close()

# function for plotting all the data points in the graph
def show_all():
  plt.clf()
  plt.subplot(1, 2, 1)
```

```
  axes = plt.subplot(1, 2, 1)
  axes.axis([0, 4, 0, 100])
  axes.xaxis.set_major_locator(MultipleLocator(0.50))
  axes.yaxis.set_major_locator(MultipleLocator(10))
  plt.xlabel("Displacement X")
  plt.ylabel("Force X")
  plt.title("Force X Vs Displacement X")
  # plt.scatter(disp1_u,force1_u)
  plt.plot(disp1_u, force1_u)
  # plt.grid(True)
  plt.subplot(1, 2, 2)
  axes = plt.subplot(1, 2, 2)
  axes.axis([0, 4, 0, 100])
  axes.xaxis.set_major_locator(MultipleLocator(0.50))
  axes.yaxis.set_major_locator(MultipleLocator(10))
  plt.xlabel("Displacement Y")
  plt.ylabel("Force Y")
  plt.title("Force Y Vs Displacement Y")
  # plt.scatter(disp2_u,force2_u)
  plt.plot(disp2_u, force2_u)
  plt.show()
  # plt.grid(True)


# starting the live plotting process
plt.ion()
fig = plt.figure(figsize=(15.5, 6))

# making list to save the data
force1 = list()
disp1 = list()
force1_u = list()
disp1_u = list()

force2 = list()
disp2 = list()
force2_u = list()
```

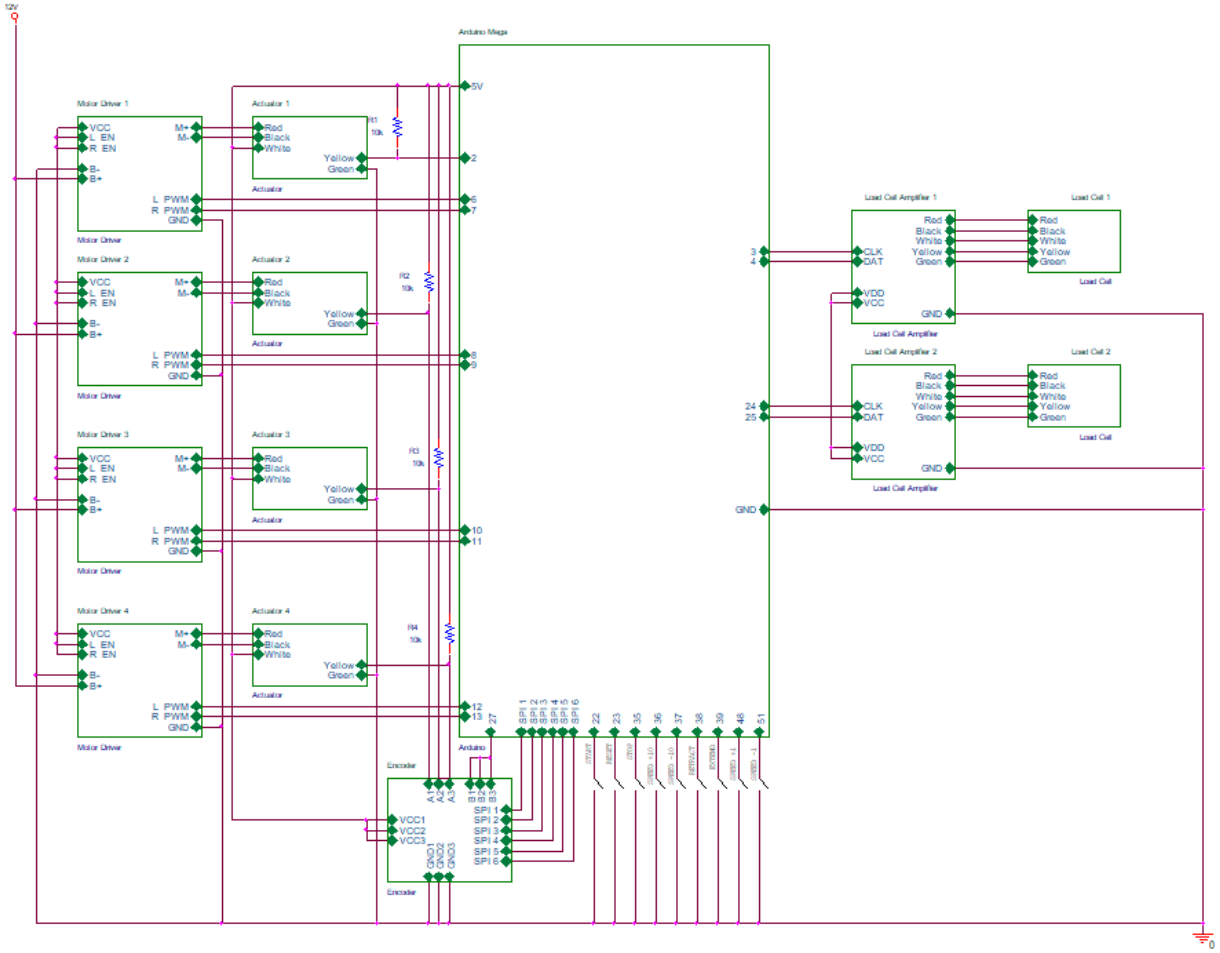
```
disp2_u = list()
speed=0.0

# making counting variables
i = 0
j = 0
k = 0

# here i used a variable called limit to call show_all90 function after 50 iterations(50 data came from the arduino)
limit = 20000

# get data from the arduino
ser = serial.Serial('Com9', 9600)
ser.close()
ser.open()

# here i used a variable which is called sample to set the number of data points show in one graph
sample = 100 # YOU HAVE TO CHANGE THIS PART

while True:

  # you have to change this as which condition youcall the function
  if k > limit:
     show_all()
     break

  # plotting the live data
  else:
     plt.clf()
     data = ser.readline()
     dec = data.decode()
     split = dec.split(",")

     split[0] = float(split[0])
     split[1] = float(split[1])
```

```
split[4] = float(split[4])
speed =split[4]

force1.append(split[0])
disp1.append(split[1])
force1_u.append(split[0])
disp1_u.append(split[1])

plt.subplot(1, 2, 1)
plt.grid(True)

if i > sample:
    force1.pop(0)
    disp1.pop(0)

axes = plt.subplot(1, 2, 1)
axes.axis([0, 4, 0, 120])
axes.xaxis.set_major_locator(MultipleLocator(0.5))
axes.yaxis.set_major_locator(MultipleLocator(10))
axes.annotate("Speed:", xy=(1.5,110))
axes.annotate(str(speed), xy=(2.0,110))

plt.xlabel("Displacement X")
plt.ylabel("Force X")
plt.title("Force X Vs Displacement Y ")
i += 1
# plt.scatter(disp1,force1)
plt.plot(disp1, force1)

split[2] = float(split[2])
split[3] = float(split[3])

force2.append(split[2])
disp2.append(split[3])
force2_u.append(split[2])
disp2_u.append(split[3])
```

```
    plt.subplot(1, 2, 2)
    plt.grid(True)

    if j > sample:
        force2.pop(0)
        disp2.pop(0)

    axes = plt.subplot(1, 2, 2)
    axes.axis([0, 4, 0, 120])
    axes.xaxis.set_major_locator(MultipleLocator(0.5))
    axes.yaxis.set_major_locator(MultipleLocator(10))
    axes.annotate("Speed:", xy=(1.5,110))
    axes.annotate(str(speed), xy=(2.0,110))

    plt.xlabel("Displacement X")
    plt.ylabel("Force Y")
    plt.title("Force X Vs Displacement Y")
    j += 1
    # plt.scatter(disp2,force2)
    plt.plot(disp2, force2)
    plt.show()
    plt.pause(0.0001)

    now = datetime.now()
    current_time = now.strftime("%H:%M:%S")
    current_time=str(current_time)

    text=""

text=current_time+","+str(force1[0])+","+str(disp1[0])+","+str(force2[0])+","+str(disp2[
0])+","+str(speed)+"\n"

    file=open(filename,"a")
    file.write(text)
    file.close()
```

$k = k + 1$

Circuit diagrams:



FAQ:

- This section will answer questions most often proposed by professors or others during presentations, as well as questions that may possibly be asked for harder to understand sections of the product.

- How do the four actuators move at the same time and speed?
    - Each actuator has its own motor driver that is controlled by the Arduino. The Arduino drives all four actuators simultaneously, at the speed and direction that the user inputs through the buttons.
- Why choose the 3mm rods over the 1mm rods?
    - The 3mm rods are thicker than the 1mm rods, so they can withstand more forces without bending. Also, the 1mm holes in the rod holder were too thin to drill reliably.
- How does the User Interface work?
    - When the user runs a test, the Arduino will print the force and displacement.These values are sent to Python using serial print .When there are values being transmitted, the graph is plotted.We are using the matplotlib and liveplotter libraries in python.

Warnings and limitations:

- Warnings:
    - Keep in mind that the rakes are sharp, so use gloves when inserting specimens and always remember to clean the rakes after use.
    - Even though all wiring is contained in the junction box attached to the tester, always pay attention to any potential exposed wires while the machine is plugged into an outlet. Electrical shock is a possibility if wires are not properly stored and are exposed where they should not be.
    - Take caution when operating the device as there are moving actuators, and even though they run at a slow speed, can catch on hair or any kind of thin fiber if caught in it while it is detracting.
    - Before running a test on a 1cm square specimen, remove the outer two rods. There is not enough space for them with this small of a specimen.
- Some limitations
    - Force applied should not exceed 80 Newtons
        - The tester cannot test anything beyond the strength of organ tissue samples, meaning no cloth or thicker non-organic material should be tested.

Deviations from SDRD:

- Emergency stop button
- Desired criteria:
    - Fluid bath
    - Overhead camera

Troubleshooting:

- Rake needs replacement due to dulling
    - Rakes can be replaced by removing the set screws and inserting a replacement, or sharpening the dull rake itself on a grindstone
- One actuator extends on start-up. Retract it before running the test so the actuators are properly aligned.
- If Arduino doesn't connect, unplug the USB and plug it back in
- If a load cell stops working properly, it needs to be resoldered.
- The EXE file uses "COM 9" so make sure you are plugging in the USB port to Com 9

Brief overview or description of each section:

- Finalized Design Requirements
    - Features the finalized SDRD associated with the system. All requirements have been updated as needed with customer approval.
- Parts list
    - This is a complete parts list, listing all parts included in the system, their price, part number, manufacturer, and quantity.
- Detailed drawings and assembly drawings
    - This section contains all necessary part drawings, as well the assembly drawings created for different subsections of the system as needed.
- Control algorithm flow charts
    - This section contains flow charts for the Arduino code and python interface. These charts describe exactly how the code runs and makes its decisions.
- Compute code
    - This section shows the whole Arduino code in order to correctly run tests. Also, contains the python code used to create the interface.
- Circuit diagrams
    - This section contains the circuit diagram of all the electrical components contained within this product. It also shows how each component is connected
- FAQ
    - This section contains questions that may be most commonly asked in regards to the system, and answers to those questions are presented.
- Warnings and limitations
    - Warnings that must be given for this product will be listed here, so that users do not cause harm to themselves or others. Also, limitations will be listed here, such as the kinds of samples the machine is capable of testing and not testing.
- Deviations from SDRD
    - Although brief, this section will list and deviations made from the SDRD, so that the customer knows which of their requirements have not been met.
- Troubleshooting
    - This last section will cover any possible problems the user may run into when operating the product, as well as troubleshooting said problems.