DEVELOPMENT OF A BIAXIAL TESTING APPARATUS

by

Amir Noorafkan

Bachelor of Engineering, McMaster University, Hamilton, Ontario, Canada (2018)

A project

presented to Ryerson University

in partial fulfilment of the

requirements for the degree of

Master of Engineering

in the program of

Aerospace Engineering

Toronto, Ontario, Canada, 2020

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A PROJECT

Development of Biaxial Testing Apparatus

Amir Noorafkan

Master of Engineering, Aerospace Engineering, Ryerson University, Toronto (2020)

ABSTRACT

Material testing is a crucial part of engineering design and development, especially in aerospace engineering as it is more cost effective to test an element or component than doing a full-scale test on the completed part. Typically, uniaxial testing is carried out to characterize a material, which is adequate for finding the properties of the material. However, these kinds of tests are inadequate for simulating the real-world loading that a component may experience in its life cycle. Therefore, this project's goal was to develop a low-cost biaxial testing apparatus using off-the-shelf components, including the Arduino Uno microcontroller ("Arduino"), stepper motors ("motors"), and load cell. This report outlines the development of the software required to operate the motors and read output value of the load cell. The Arduino code used to control the motors was developed using open-source code available on GitHub and the *Stepper* library, which contains the required functions for controlling the motors. The Arduino code can be used to determine the strain rate of up to $11\ mm/min$, as well as the type of loading (tension or compression) along each axis.

ACKNOWLEDGMENT

I would like to thank my supervisors Dr. Hamid Ghaemi and Dr. Kazem Fayazbakhsh of the Department of Aerospace Engineering at Ryerson University. Their continued support and guidance helped me advance my knowledge and understanding of biaxial testing methods, testing standards, and test equipment development. Their continuous feedback has been incredibly significant in steering my project work in the right direction.

Finally, I would like to thank my family for their understanding, support, and encouragement throughout the course of this project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF APPENDICES

1. Introduction

Advances in material sciences manufacturing has opened the door for creating new material as well as incorporating them into new designs. These advances have made their way into aerospace applications, with aircrafts such as the Airbus A350 having 70% of its composition consisting of advanced materials - combining carbon composites, titanium and advanced aluminum alloys [1]. In order to incorporate new materials to create better aircrafts, first the material must be tested in real life conditions to understand its behaviour before large amounts of resources are allocated to designing and manufacturing a new aircraft.

Uniaxial tests, for example shear and tensile tests, are enough to provide the data required for characterizing a material. However, these tests are inadequate for simulating real life conditions that a given structure will experience throughout its service life. In many cases the loads on a component are not simple stresses acting uniaxially, but they are combined loads - *i.e.* biaxial loads. Examples of components that experience combined loading are fuselage panels in aircrafts, as well as sheet metal forming in manufacturing. Hence it is important to understand the stress-strain relationship of components that are subjected to combined loading.

This project attempted to develop a biaxial testing apparatus using low-cost and off-the-shelf equipment for testing primarily 3D printed specimens.

2. Literature Review

Biaxial load tests are common for testing fabrics and biomedical specimens, as these materials often experience combined loading in real life environments. The increasing demand for biomedical and other types of material is one of the main drivers for the development of biaxial testing rigs in the biomedical industry and for research purposes [2].

A study conducted by the Center for Memory and Recording Research at the University of California San Diego, employed biaxial testing alongside uniaxial testing to study the nonlinear material response of 3D printed photopolymer material. Their biaxial tests demonstrated isotropic and hyperelastic behaviour of the photopolymer material, and due to the large number of deformation states allow for a unique identification of the Mooney-Rivlin coefficients [3] - a model used for the large nonlinear strain behaviour of incompressible material [4].

2.1. Stress

In engineering mechanics, stress has the units of Pascal (N/m^2)derived from external forces acting over a cross-sectional area of a structure. A structure can experience two main types of stress [5].

- Normal stress: the force acting on the structure is perpendicular (or normal) to the cross-sectional area,
- Shear stress: the force acting on the object is parallel to the cross-sectional area.

Depending on the direction of the normal stress, the structure may be under compression or tension. Other stress states can be achieved from the combination of normal and shear stresses along different axes. The different types of uniaxial mechanical stress is illustrated in Figure 1.

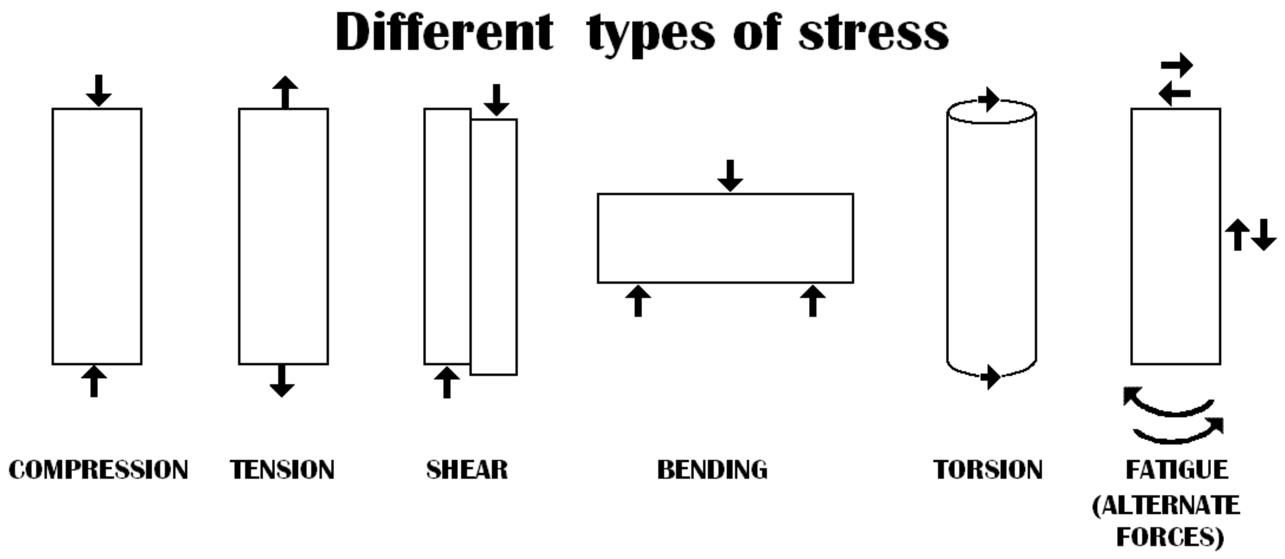


Figure 1 - Types of mechanical loading on a material [6]

Biaxial stress (plane stress) occurs when forces applied on an object coincide on the same plane, that is, the forces normal to that plane are much smaller than the planar stresses and negligible [7]. The biaxial testing apparatus was designed to reproduce the biaxial stress states illustrated in Figure 2

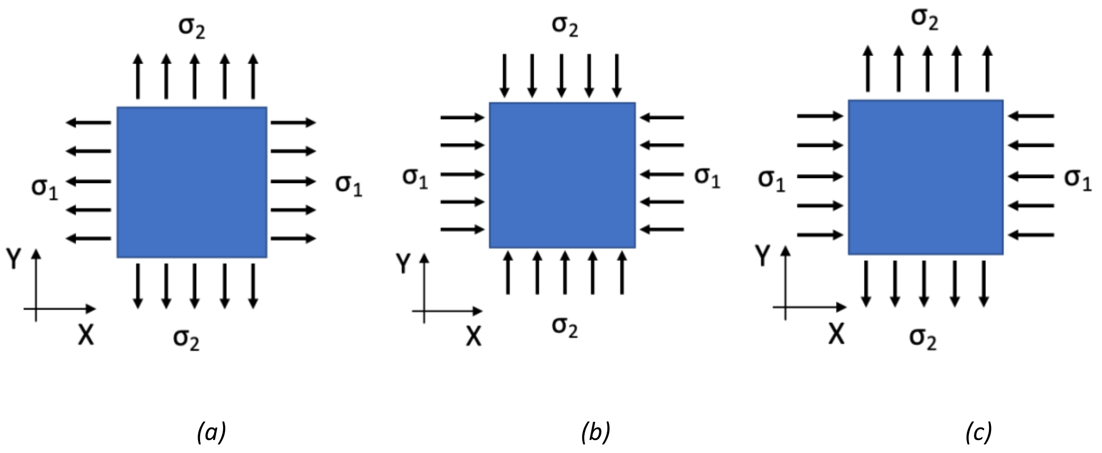


Figure 2 - a) Tension in both X and Y axes, b) Compression in both X and Y axis, c) Tension in X axis and compression in Y axis

An example of plane stress can be found in pressurized vessels where the vessel's thin walls experience hoop stress and longitudinal stresses, as illustrated in Figure 3. In this simple model the effects of the curvature of the walls of the vessel are neglected. Thin-walled tubes subjected to tension or compression in combination with torsion or internal/external pressure for creating a biaxial stress state is used by some researchers; however, the downside of this method is that the test is not purely biaxial due to the radial stress gradient of the tube [8].

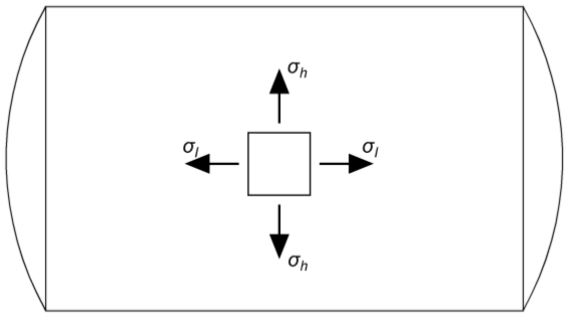


Figure 3 - Biaxial loads on thin walls of a cylinder or pressurized vessel [9]

For each stress state, forces can be applied to the specimens in two configurations as shown in Figure 4. For a biaxial load where the structure or object is constrained on one side and forces are only applied on one side along each axis (Figure 4.a), as the specimen deforms the center will shift. Forces applied evenly on all four sides of the sample (Figure 4.b) can prevent the center from shifting away from its initial position [10]. This is an important feature for a biaxial testing apparatus, as the center will provide a frame of reference for measuring strain.

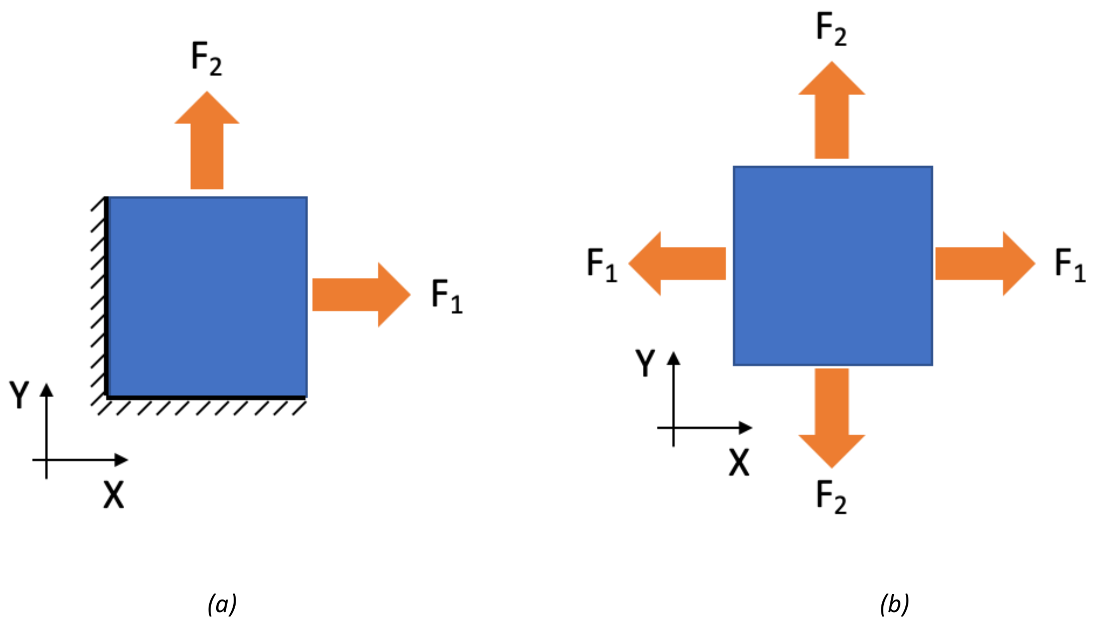


Figure 4 - a) Force applied on one side of specimen with other side anchored, b) Force applied on both sides of specimen

Another type of stress is isotropic stress that occurs when equal tensile or compressive forces are applied along three axes of an object, as shown in Figure 5. An example of this would be hydrostatic loading, where an object submerged under water experiences pressure evenly along all sides.

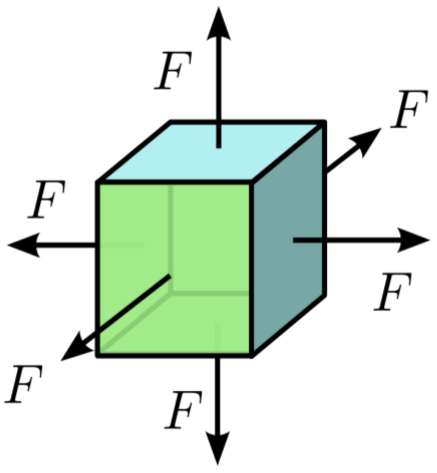


Figure 5 - Isotropic stress, equal tensile forces applied along all axes [11]

2.2. Shear

Shear stress is the component of stress parallel to its cross-section, which promotes the deformation of the material along its slippage planes parallel the direction of the stress [12]. Figure 6 shows an illustration of the shear stress and the deformation it causes on a square shaped material that deforms into a parallelepiped.

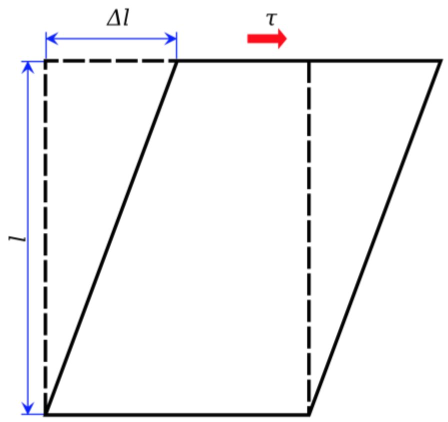


Figure 6 - Illustration of shear stress on an element and the resulting deformation [13]

The stress states covered so far are considered simple stress states as they do not account for shear stress induced on each surface of the object as a result of tensile or compressive forces on the object. In reality, a structure experiences combined stress, that is some combination of multiple forces with each acting in a different direction on an object or point. The schematic in Figure 7 shows how these forces on an object can be broken down, and from which it can be seen that on each surface there exists a normal force perpendicular to that surface, as well as two shear components parallel to the surface.

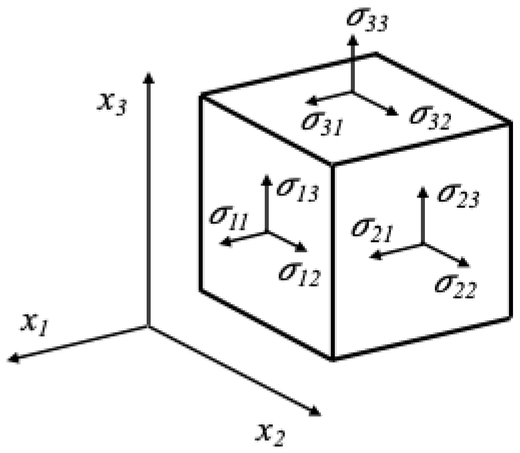


Figure 7 - General stress state, illustrating all stresses and strains on an object

Since there are three stress components on each surface, a stress tensor with nine stress components known as the Cauchy stress tensor can be created for all the forces acting on an object or point, describing its stress state [11] as shown in Figure 8.

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix} \equiv \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix} \equiv \begin{bmatrix} \sigma_{x} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_{y} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_{z} \end{bmatrix}$$

Figure 8 - Cauchy stress tensor describing stress state of a given object or point [13]

2.3. Strain

Strain describes the tendency of a material to change its formation from its original shape to a new configuration when a force is acting on the material [14], it is the ratio between the material's original size and its deformed size, which makes strain unitless. Strain is related to the magnitude of the force being applied to the material and the material's stiffness. Strain, or deformation, can occur from external forces acting on the material, or other sources such as change in temperature and electromagnetic forces [14]. Similar to types of stress, there are two types of strain: normal strain, and shear strain. Normal strain takes place when the deformation is in response to a normal stress, and shear strain occurs when the deformation is in response to a shear stress. For example, a beam under tension will elongate relative to the applied tension, as shown in the illustration in Figure 9, displays normal strain.

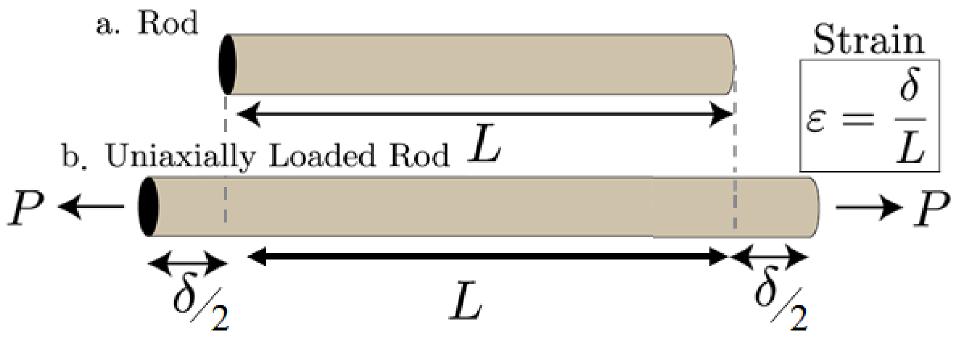


Figure 9 - Normal strain on a rod under tensile loading [15]

In the case of biaxial stress, or *plane stress*, although the stress along the thickness direction of the material is negligible, the material will still show strain along the thickness direction. With this, the strain of the thickness of the material can be calculated using Equation 1 [7].

$$\epsilon_3 = -\nu\frac{(\sigma_1+\sigma_2)}{E} \tag{1}$$

Where

ϵ_3 - strain along thickness

ν - Poisson's ratio

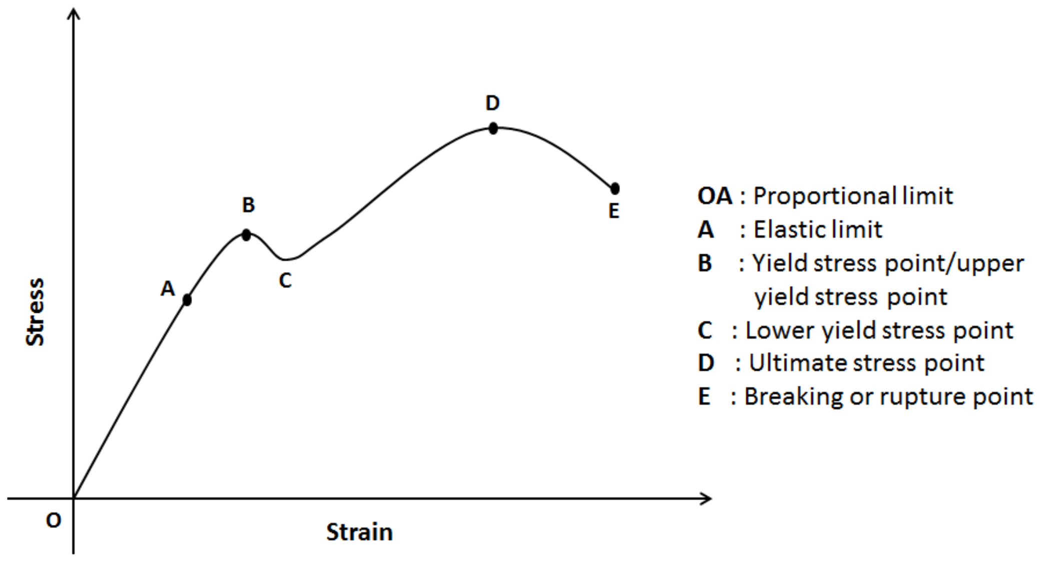
σ_1, σ_2 - stresses along axes 1 and 2

However, if the material is constrained along the thickness direction (*i.e.* $\epsilon_3 = 0$), then a stress along the thickness, σ_3, will develop. Since there is only strain across one plane and stress along all three directions, this case is known as *plane strain.* The stress along the thickness direction can be calculated from Hooke's law, using Equation 2 [7].

$$\sigma_3 = \nu(\sigma_1 + \sigma_2) \tag{2}$$

2.4. Stress-Strain

From what has been gathered so far, there exists a relationship between stress and strain, and a curve describing this relationship can be created for any material through testing. Figure 10 shows a typical stress-strain curve with labels for each noteworthy point on the curve. When a material undergoes elastic strain up to its elastic limit, the material can return to its original shape without any damage to the material once the applied force is removed. If a large enough force is applied to a material, the strain will go beyond the elastic limit and plastic strain will begin. Plastic deformation is permanent, and the material will stay deformed even after the load has been removed. Plastic deformation will occur up to the point where the material cannot withstand any more loads, necking begins, and cracks start to develop until the material fails.



D
B
E
A
C
Stress
O
Strain
OA : Proportional limit
A : Elastic limit
B : Yield stress point/upper yield stress point
C : Lower yield stress point
D : Ultimate stress point
E : Breaking or rupture point

Figure 10 - Typical stress-strain curve with labels [16]

2.5. Poisson's Ratio

A material with uniform properties in all directions is *isotropic*. At macroscopic scales, most metals and ceramics are approximately isotropic, since they are composed of randomly oriented crystal grains. A bar of isotropic material, with initial length L_i and initial diameter of d_i (Figure 11), under axial stress σ_X will exhibit an axial strain of ε_X calculated using Equation 3.

$$\varepsilon_x = \frac{L-L_i}{L_i} = \frac{\Delta L}{L_i} \tag{3}$$

Where L is the length of material under load, experiencing strain.

Similarly, the directions perpendicular to the stress will exhibit lateral strain, which can be calculated using Equation 4.

$$\varepsilon_y = \varepsilon_z = \frac{d-d_i}{d_i} = \frac{\Delta d}{d_i} \tag{4}$$

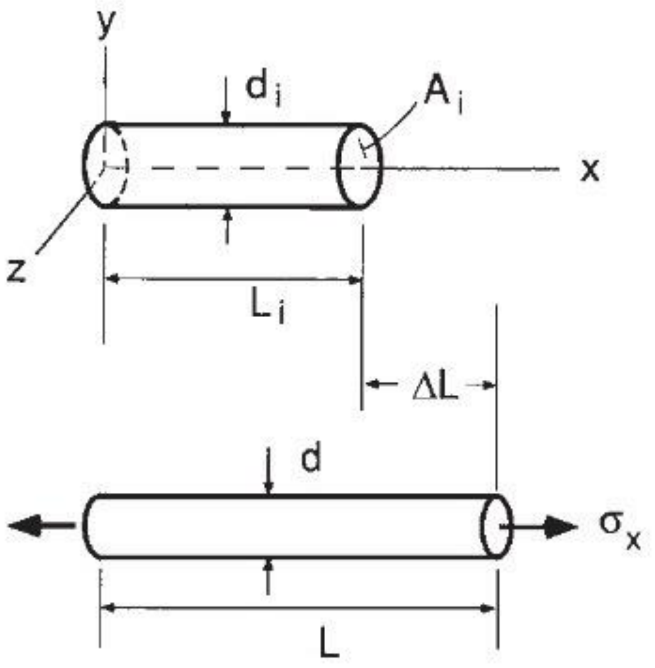


Figure 11 - Isotropic bar under axial stress [27]

When the bar is in tension along the X-axis, the diameter decreases, which means lateral strain is negative, and positive when the rod is under compression. A material is said to be *linear elastic* if the strain returns to zero after load is removed. In a linear elastic case, two relationships can be established, which are the *elastic modulus* shown in Equation 5, and *Poisson's ratio* shown in Equation 6.

$$E = \frac{\sigma_x}{\varepsilon_x} \tag{5}$$

$$v = -\frac{axial\ strain}{longitudinal\ strain} = -\frac{\varepsilon_y}{\varepsilon_x} \tag{6}$$

-Poisson's ratio in most cases falls around 0.3 and cannot go beyond the range of 0 to 0.5, unless it is a unique case. A negative Poisson's ratio would indicate lateral expansion when the material is under tensile loading, and a value higher than 0.5 would indicate the volume of the material is decreasing under tensile load, which are unlikely scenarios [27].

2.6. In-Plane Biaxial Test Machines

One of the first available biaxial testing machines mentioned in literature was a device developed by Rudolf Haas in 1913 and used for testing fabrics that were utilized in the construction of zeppelins [18] [19]. The device constructed by Haas, shown in Figure 12, applied force symmetrically on the samples, which is a common and cost-effective method for producing biaxial loads on a sample [20].

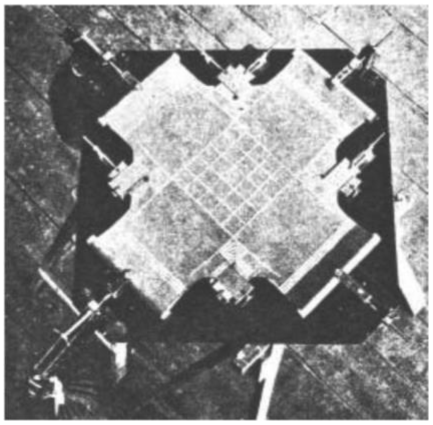


Figure 12 - First biaxial testing machine found in literature, developed by Rudolf Haas in 1913 [21]

An example of the biaxial testing machine is the University of Minnesota's device shown in Figure 13, which is used for performing mechanical testing and property analysis of natural and bioartificial soft tissues [20].

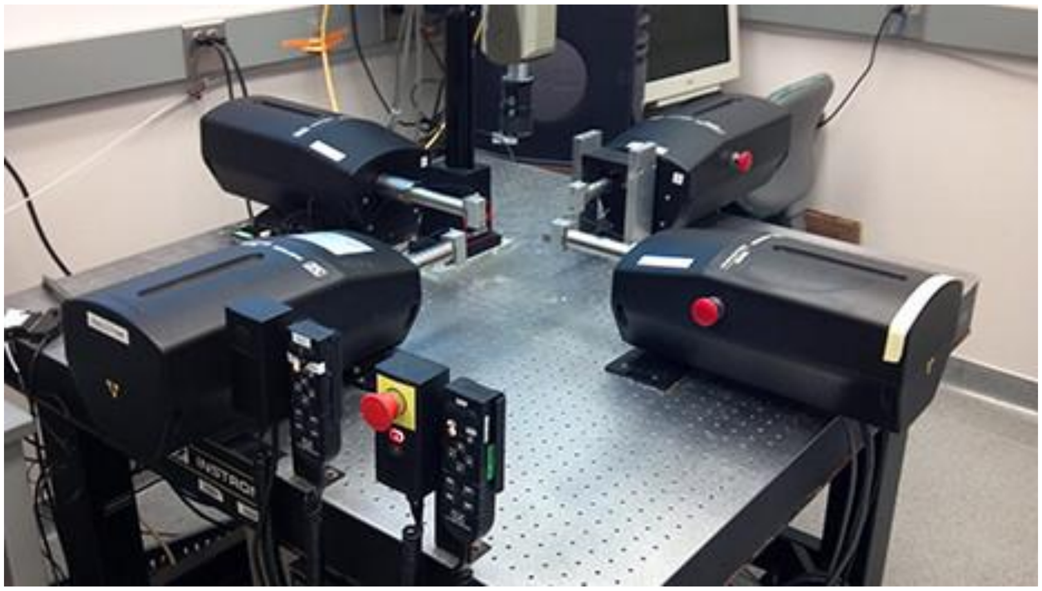


Figure 13 - University of Minnesota department of Biomedical Engineering Instron-Sacks planar biaxial testing system [20]

Companies such as Zwick Roell and TestResources Inc. are manufacturers of biaxial testing machines, with force applying capabilities ranging from 100 N for biomaterials all the way up to 2 MN for metals and ceramics [18][19]. The department of Computational Mechanics at the Technical University of Munich uses a biaxial testing machine from Zwick Roell to perform research on material modelling of biological tissues, such as aortic walls, under high strain [19].

With the growing interest in studying biaxial loading on components, and the decrease in cost of computation with microcontrollers has created an opportunity for smaller research groups to develop their own biaxial testing apparatus using off-the-shelf components. The department of Architecture and Built Environment of the University of Nottingham developed a cost-effective biaxial testing machine architectural membrane, using a Raspberry Pi microcontroller to log the data generated from tests. The researcher used four S type load cells (model Phidget CZL301), which can handle loads of up to 5 kN. The loads are applied manually through a threaded bar, but the researchers stated their next development will use the

Raspberry Pi to control actuators to apply the required force. Three slide potentiometers measure elongation on the sample, which along with data on the force applied are simultaneously recorded by the Raspberry Pi. Additionally, the microcontroller measures and records temperature and humidity through four channels and has two sensors to measure pressure. The researchers use force and elongation data to create the stress-strain response curve of a given material [10].

2.7. Specimen Shape

Based on reviewing the literature, the best way to induce a biaxial stress state in a sheet of material is to apply in-plane loads along the perpendicular axis to the arms of a cruciform shaped sample [8]. Furthermore, one of the earliest test procedures for plane biaxial tensile testing was developed by the Membrane Structures Association of Japan (MSAJ/M-02:1995), in which they describe the specimen shape to be a cruciform [21].

However, there are many ways to create a cruciform shaped specimen, and there are instances in literature where cruciform specimens were created empirically. Demmerle and Boehler [22] developed a finite element model for optimizing the geometry of flat specimens for biaxial testing for isotropic and anisotropic material, using tests conducted on cold-rolled Cu-Zn alloy. They outlined three criterions that must be taken into consideration when applying their finite element model:

- Central testing area must experience homogeneous stress and strain distribution,
- The nominal stress values, derived from dividing each of the applied loads on the arms by their corresponding cross-sectional area, must be compatible with with the stress values in the test region,
- Highest stress level and initial yielding must occur in the central test region, and there must not exist any stress concentration outside of the test region.

For their finite element model, they considered only ¼ of the specimen due to symmetrical loading and specimen shape, shown in Figure 14.

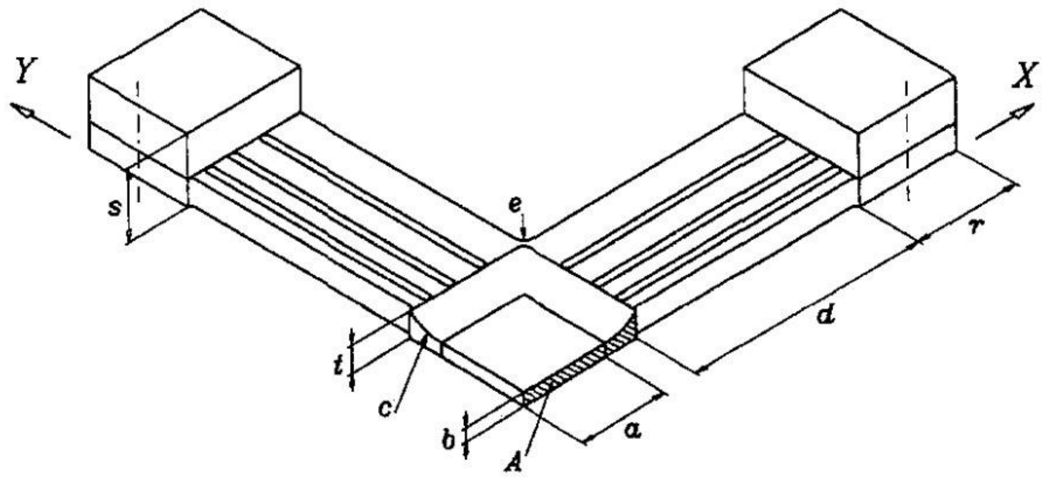


Figure 14 - Geometrical parameters for the finite element model used for optimization of specimen shape [8]

Based on Demmerle and Boehler's initial numerical stress analysis, the number of slots within the width of the specimen's arms was determined to be seven, meaning there are seven slots on the arms of the cruciform shaped sample for the testing apparatus grips to clamp on to and locked in place. These slots are connections for the biaxial testing machine grips to attach to.

With an odd number of slots in the sample's arms, there will exist one slot at the center of the arms of the cruciform sample, which was found to reduce the stress level in that region. The final design of the specimen they derived is shown in Figure 15. It should be noted that the test area in the center has a smaller thickness than the arms of the sample. For more details on this model please refer to Appendix A.

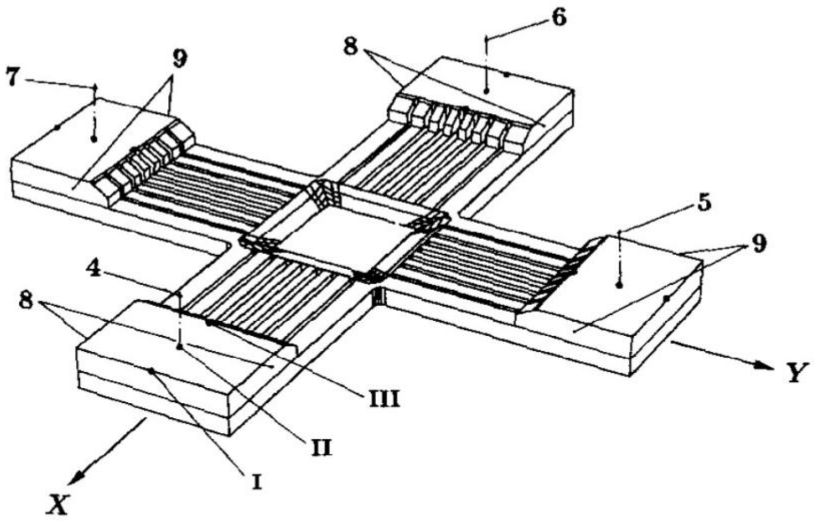


Figure 15 - Optimal cruciform design of specimen based on the finite element model created by Demmerle and Boehler [22]

A study conducted by a NASA contractor for the Langley Research Center [23] identified that although cruciform specimens are ideal for biaxial testing, they too present some complications. When uniaxial testing is done, the stress on the sample is calculated using the applied load and the cross-sectional area of the specimen. However due to increased complexity of the cruciform sample, numerical analysis would be required to calculate the stress from applied loads. Other things to consider when using cruciform shaped samples:

- specimen alignment,
- size and uniformity of the biaxial stress,
- stress concentrations outside the test region,
- and stress redistribution after localized yielding.

Regarding the size of cruciform shaped samples for biaxial testing, ISO 16842:2014 standard for measuring the stress-strain curve of metallic materials for sheet and strip, outlines the thickness of the specimen must be 0.1 mm or more and 0.08 times or less than the cruciform arm width of the specimen [24].

Lastly, it is important to note that the cruciform shape and biaxial testing in general is best suited for isotropic material, with similar properties along all directions of the specimen.

3. Setup & Design of Biaxial Testing Apparatus

The biaxial testing apparatus was designed to apply tensile, compressive, or mix mode loading along two perpendicular axes on one plane. The apparatus has two arms with grips along each axis to hold the specimen in place and apply the desired loads on to the specimen. The arms of the machine consist of grips attached to a load cell on a rod attached to a machined aluminum U-support with a ball screw element incorporated into the opposite end, as shown in the CAD model in Figure 16. The motors are linked to a stationary threaded rod passing through the ball screw on the U-support, with slide rails mounted to L-brackets to allow for balanced and free movement of the arms - CAD model shown in Figure 17. The full arm and grip system is shown in Figure 18.

The arm and grip components are bolted to a 1.2m by 1.2m steel plate with thickness of 100 mm, which is strong enough to withstand any bending or deformation that the loads during testing can induce. The 3D CAD model of the biaxial testing apparatus is shown in Figure 19. 3D CAD models were created by graduate student Giovanna Saccaro.

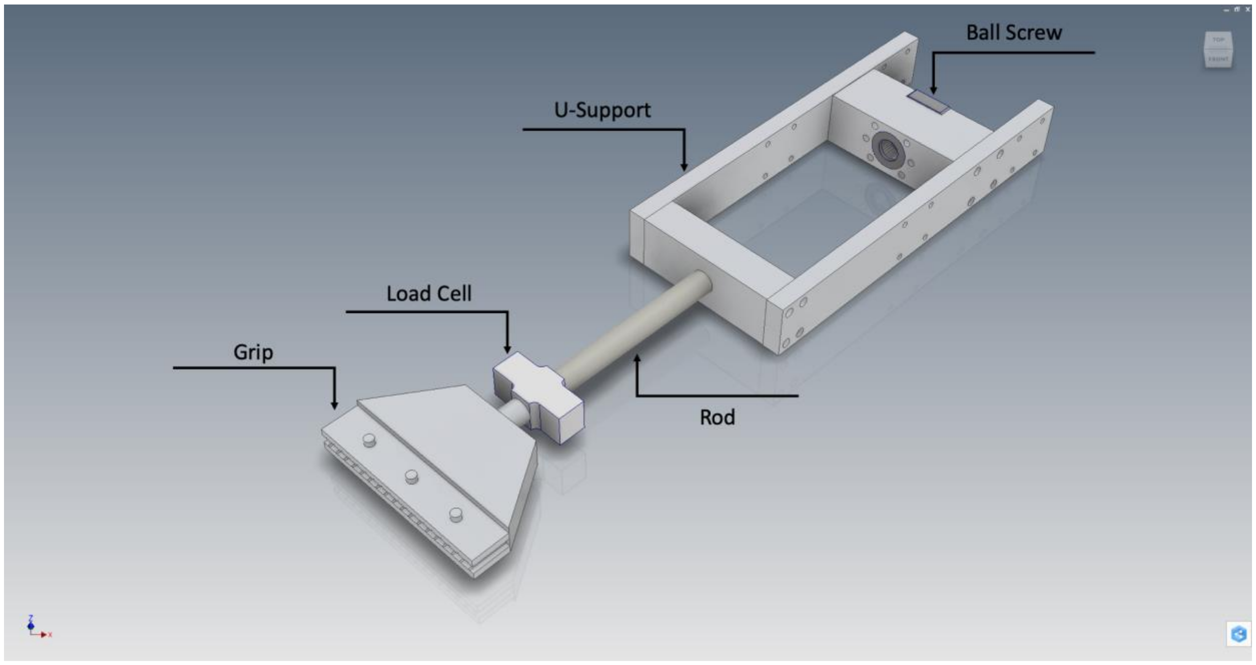


Figure 16 - 3D CAD model of the arm and grip system of the biaxial testing apparatus

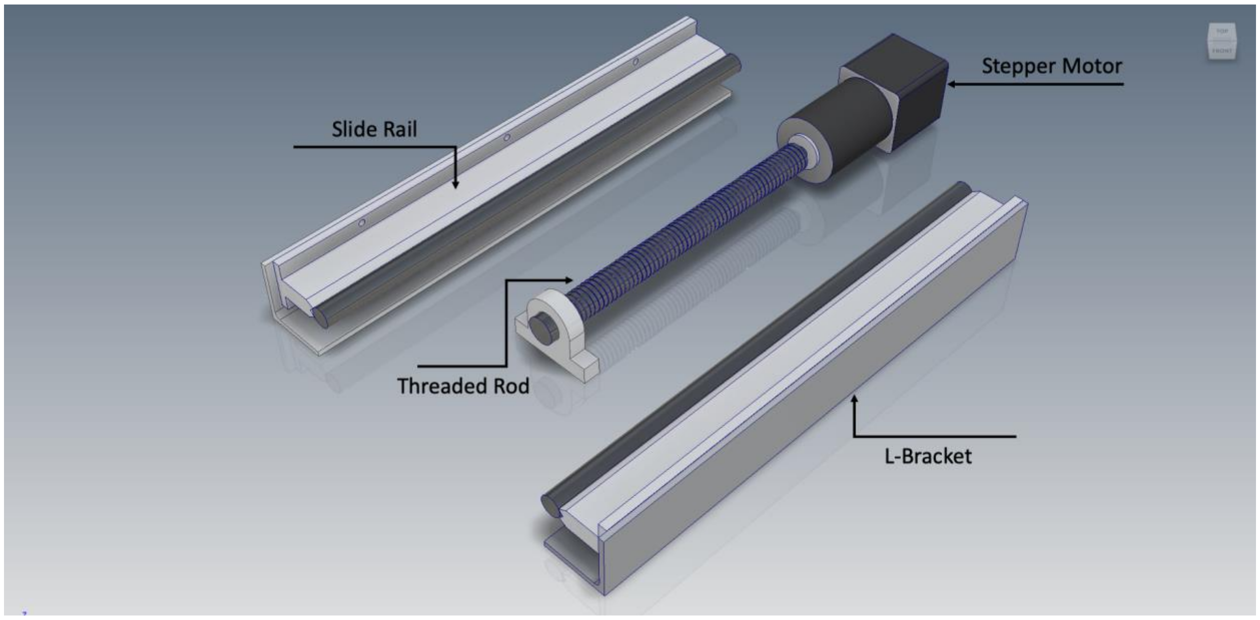


Figure 17 - 3D CAD model of the side rail and motorized

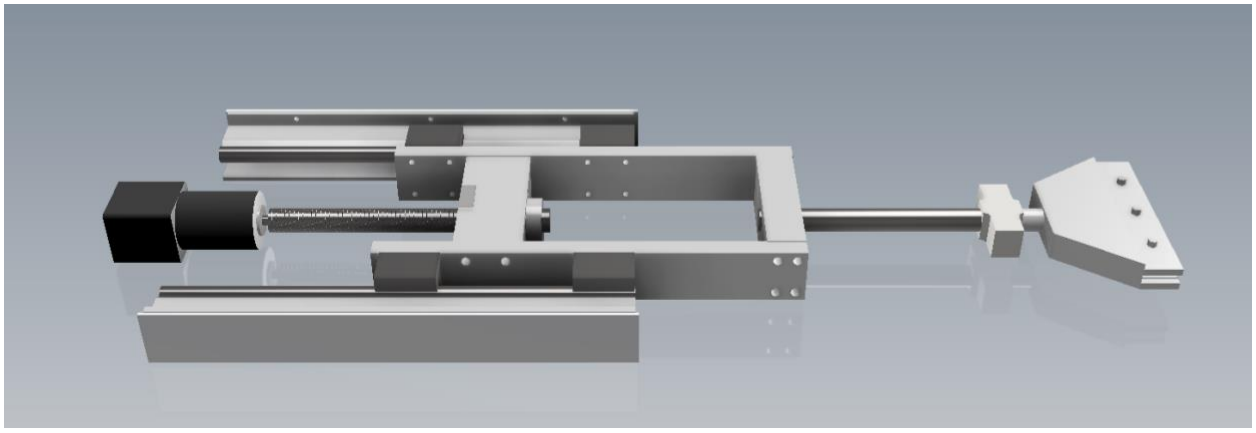


Figure 18 - 3D CAD model of full arm and grip system of biaxial testing apparatus

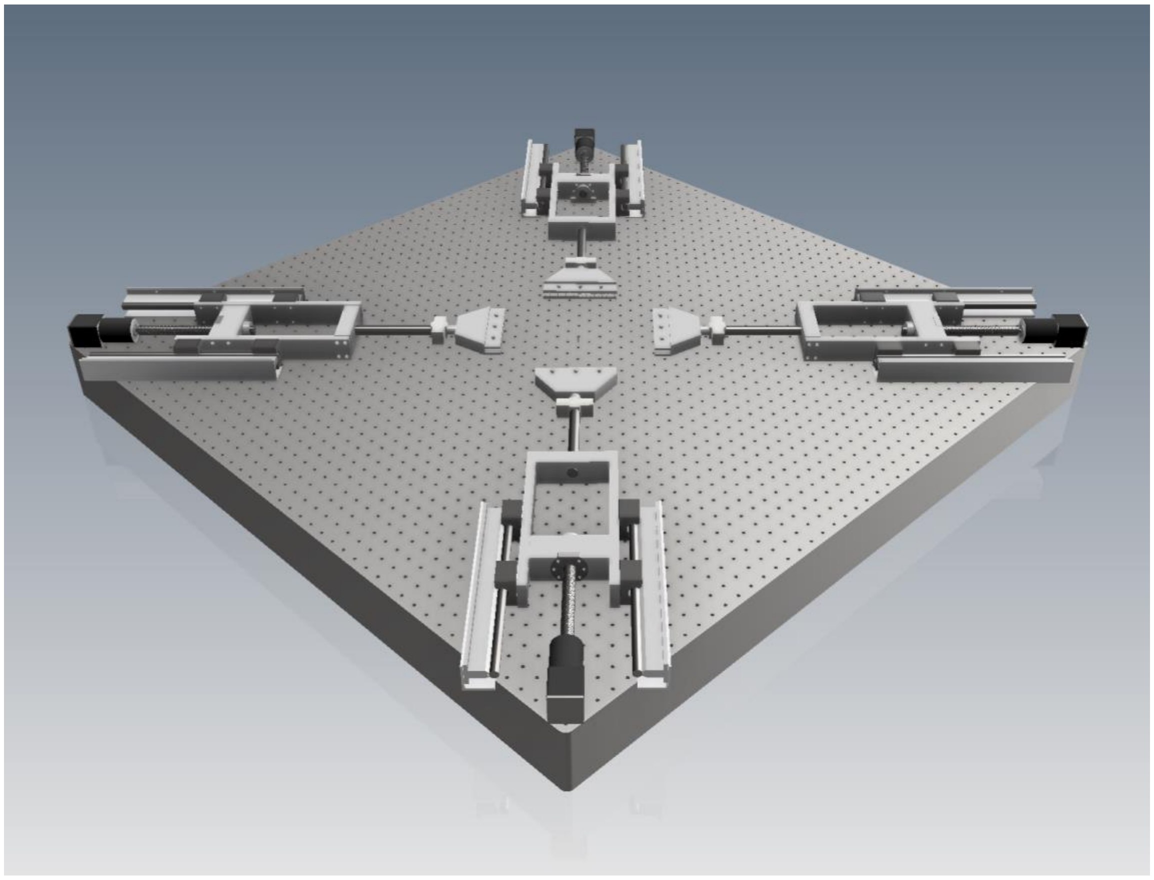


Figure 19 - 3D CAD model of biaxial testing apparatus

The operator of the apparatus can determine the strain rate and type of testing (tension, compression, or combined). During testing, the apparatus will record loads on each axis until it detects a major drop in loading, which can be used as an indication of failure of the specimen and stop the test.

3.1. Hardware

The design of the testing apparatus incorporates lead screws, rotated by the stepper motors. Stepper motors are used to generate the required torque and rotate the lead screws at the desired speed and direction, to induce tensile or compressive loading on samples. Linear motion is achieved by passing the rotating lead screw through a ball screw, similar to the one shown in Figure 20.

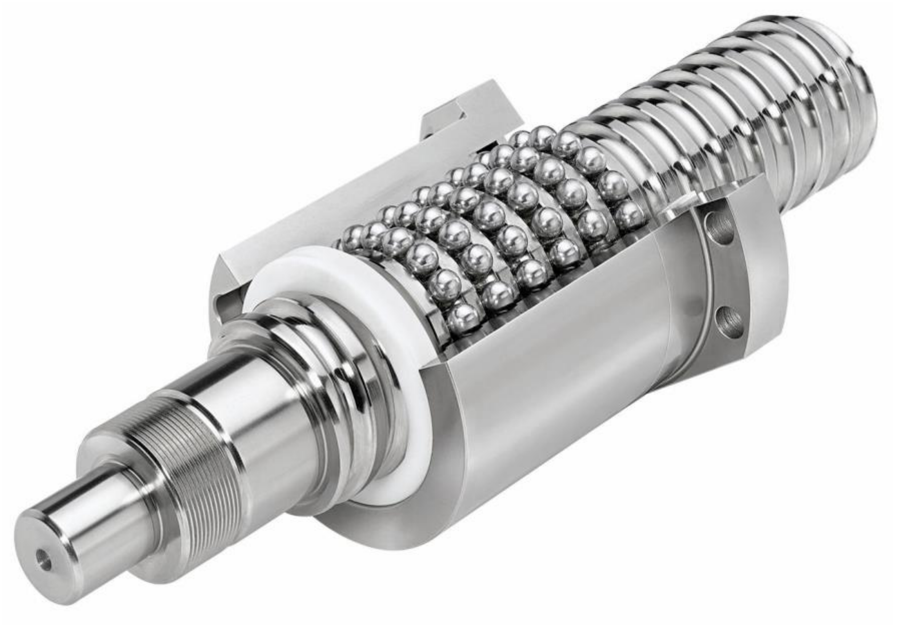


Figure 20 - Lead screw passing through a ball screw [25]

The ball screw is incorporated at the base of the U-support component of the arms, as shown in the 3D CAD model in Figure 21.

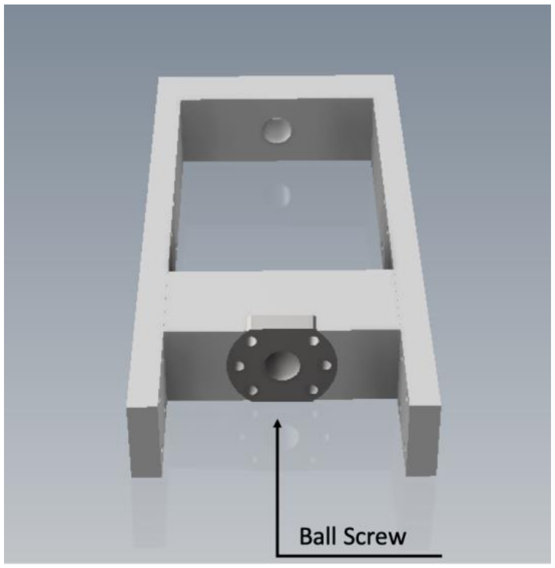


Figure 21 - 3D CAD model of U-support component with ball screw

Stepper motors have multiple coils that energize in sequence to rotate the motor, which leads to precise speed and positioning control and makes steppers motors ideal for the bi-axial testing apparatus.

The stepper motors are connected to a digital stepping driver, which in turn is connected to a power supply. The digital stepper driver is then connected to an Arduino Uno that gives instructions regarding the speed and direction of rotation. The rotational speed of the stepper motors determines the strain rate. Since the lead screw pitch is 5 mm apart, one full rotation of the stepper motor will result in a 5 mm strain on the specimen.

To measure the applied loads, a 1000 lbs. capacity load cell is incorporated in the arms, located before the grips, as shown previously in Figure 16. The load cell is first connected to the load cell amplifier, then the amplifier is connected to the Arduino. The load cell requires a steady power supply of 10 V, and it cannot get its power from the Arduino. The minimum voltage that the power supply connected to the stepping driver can provide is 24 V. Therefore, the load cell was connected to a separate power supply. The diagram in Figure 22 illustrates the Arduino circuit and connections with other electrical components. Actual pin connections may differ from the diagram.

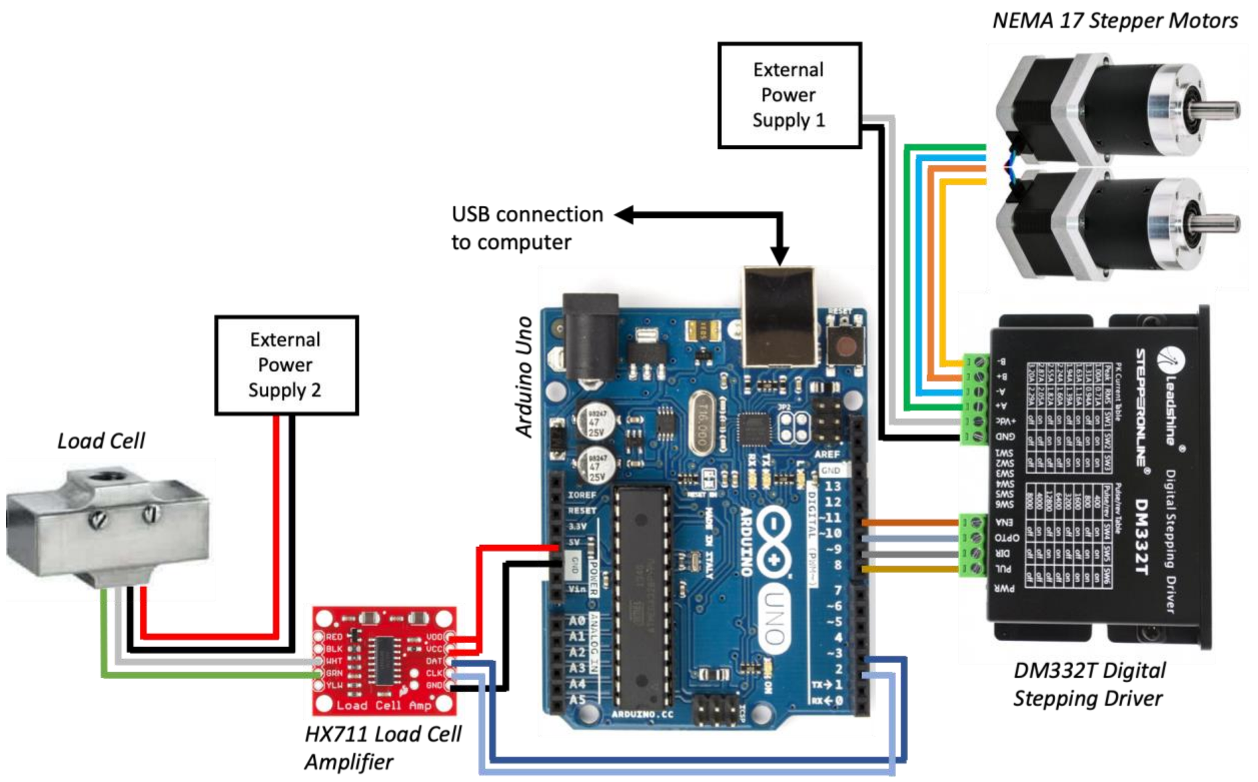


Figure 22 - Illustration of circuit connection between Arduino Uno and other components

3.2. Software

The Arduino Uno programming language is a set of C/C++ functions that are called from the code written in the Arduino Integrated Development Environment (IDE).

In the IDE, firstly the relevant libraries must be imported for use by the different components. Then the pins which the stepper driver is connected to are determined and any other variable (integer, string, float) is defined. The code contains two sections: “void setup ()” and “void loop ()”. In the setup section, the mode of each pin is defined, e.g. OUTPUT or INPUT, as well as any other text required to show in the Serial Monitor. The serial monitor is used for entering data into an Arduino program, as well as viewing data output by the Arduino program. The “Serial.begin” command initiates the serial data transmission rate, in bits per second (baud), between the Arduino and the stepper driver. Only the baud available in the serial monitor drop down menu at the bottom of the window was used and can be accessed through the serial monitor by CMD+SHIFT+M on Mac, and CTRL+SHIFT+M on Windows.

The final code is a combination of multiple open source scripts for the Arduino Uno, which were adjusted to meet the requirements for this project. The code for adjusting strain rate and load type (tension or compression) can be seen in Appendix C.

3.3. Stepper Motors

The loop section of the code is where the main functions that will control the stepper motors are placed. It starts with a “while()” loop to take input from the serial monitor that the user enters, and activates a specific function. The direction of rotation is determined by “1” (clockwise) or “2” (counterclockwise) or “0” (stop). The next input separated by a comma is the strain rate (or how fast the motors will rotate). The maximum strain rate with the current setup is approximately 11 mm/min. The layout of the commands to the serial monitor must be as follows:

[rotation dir. X-axis] , [strain rate X-axis] , [rotation dir. Y-axis] , [strain rate Y-axis]

The way the code was set up right now it cannot run both pairs of stepper motors for each axis simultaneously. This is because of how the Arduino sends signals to the stepper driver, and the bandwidth limitation of the Arduino. In other words, only a certain number of signals can be sent and received at a time. Currently, one pair would run for a certain amount of time, then the other pair would run. To circumvent this issue, the code was tweaked to only make each pair of motors rotate 100 steps at a time. Since the motors have a total of 20,000 steps each, running them for 100 steps takes a very short amount of time, so the motors seem as if they are running simultaneously, when in reality they are essentially taking turns rotating for 100 steps. The lead screw teeth are 0.3707 mm apart, which means every rotation will result in 0.3707 mm movement along the axis. The code uses this measurement and the strain rate entered by the user to adjust the rotational speed of the motors. The time it takes for each set of motors to complete 100 steps can be calculated as follows;

(100 steps)/(20,000 steps per rotation) = **0.005 of one full rotation**

(0.005)x(360 degrees per rotation) = 1.8 degrees per turn for each set of motors

With the lead screw teeth at 0.3707 mm apart, 1.8^o rotation of the stepper motor will result in **1.85x10^{-3} mm per turn**. In the code for this program, the variables *strainVar1* and *strainVar2*, for first and second axis respectively, takes the user input for the strain rate of each axis, *strainRate1* and *strainRate2*, and converts to the number of full rotations of the stepper motors. This is done by dividing the *strainRate* from user input by 0.3037, which is the travel distance per rotation of the lead screw - *i.e.* one rotation of the lead screw results in *0.3037 mm* linear travel. An example of this operation is shown below for two cases;

Case A

- User input: strainRate = 4.2 mm/min
- Program: strainVar = (4.2 mm/min) / (0.3037 mm/rotation) = 13.83 rotations/min

Case B

- User input: strainRate = 2.3 mm/min
- Program: strainVar = (2.3 mm/min) / (0.3037 mm/rotation) = 7.57 rotations/min

In Case A the strain rate entered by the user is higher than the strain rate entered for Case B, so for the stepper motors to achieve this higher strain rate they must complete more rotations per minute in Case A than for Case B. So now we can calculate the time each set of motors runs for in their turn;

Case A

t = (0.005 of a full rotation of the motor per turn) / (13.83 rotations/min) = 3.62x10^{-4} min = 0.0217 seconds (or **21.7 milliseconds**)

Case B

t = (0.005 of a full rotation of the motor per turn) / (7.57 rotations/min) = 6.61x10^{-4} min = 0.0396 seconds (or **39.6 milliseconds**)

This means that in Case A each set of motors runs for *21.7 ms* during their turn, and in Case B each set of motors runs for *39.6 ms* during their turn. So, the 100 steps per turn mentioned earlier translates to different durations that each set of motors runs for based on the strain rate that the user enters.

Although the motors can run pseudo-simultaneously, they cannot have different strain rates because of this bottleneck. For example, following the layout of commands to the serial monitor, if the user enters the values 1, 7.6, 2, 10.4 into the serial monitor, the result is X-axis rotating clockwise with strain rate of 7.6 mm/min, and Y-axis rotating counterclockwise with the same strain rate of 7.6 mm/min instead of the 2 mm/min that is entered.

3.4. Load Cell

After connecting the load cell to the power supply, amplifier, and to the Arduino, calibration was done using known weights. The calibration involves running the source code for the load cell amplifier provided by the seller of the amplifier, and importing the open source Arduino library to interface the Avia Semiconductor HX711 Analog-to-Digital Converter for reading load cell data [26]. The documentation for this library is comprehensive and includes a full step-by-step guide on how to conduct the calibration.

The code for calibrating the load cell and amplifier works by measuring the voltage change as a result of force applied on the load cell. During the initial run of the calibration code, no weight must be placed on the load cell to allow for stabilization of the readings that display on the serial monitor. Once the readings are stabilized, a known weight was placed on the load cell, which will cause fluctuations to the reading. Using the "+" (plus) or "-" (minus) inputs to the serial monitor the reading can be adjusted until the correct value is showing. The weight was then removed, and another weight of known value was placed on the load cell to test the calibration. The calibration code uses an arbitrary value for the calibration factor, so another way to calibrate the readings is to manually adjust this factor in the source code and placing known weights on the load cell to see whether or not the value entered is correct or not. This is useful if the same load cell and Arduino configuration is being replicated, as the user can simply enter the newly discovered calibration factor to the source code and no longer require calibration for the new setup.

It is important to use masses with known weights in order to properly calibrate the load cell using this calibration program. The weights used for calibration in this setup were 1 lbs., 5 lbs., and 10 lbs., with the calibration complete larger heavier weights (combination of known masses) were placed on the load cell to test the calibration. The readings from the load cell showed a variation of up to 0.2 lbs. for each weight.

Furthermore, the weight units of the source code are in pounds, but can easily be changed to kilograms, which will require reconfiguration of the calibration factor.

4. Results

Based on the literature survey, it is possible to develop a robust biaxial testing apparatus at a low cost compared to industrial or scientific biaxial testing machines.

Three Arduino programs were developed for this project. The first was developed to control the stepper motors, one program for calibrating the load cell, and another for reading and displaying the load cell data.

First Arduino program: strain rate and load type control

Two stepper motors are used for each axis, that is one pair of motors per axis. In each pair, the wires of the motors are joined so that one set of instructions by the Arduino can operate both motors. Each pair is connected to a digital stepping driver, which means two drivers are connected to the Arduino Uno. Using the code in Appendix C, the rotation speed and direction of the motors is controlled. To operate the motors, the user should first open the Arduino serial monitor (found under the “Tools” tab in the Arduino IDE), then upload the program to the Arduino Uno. After which the program will give instructions to the user on how to enter the desired strain rate and load type (tension or compression). When these values are entered, the program will use the strain rate to calculate the rotation speed of the motors and send the appropriate instruction to the motors to start rotating in the direction which correlates to compression or tension loading, as entered by the user. The motors can apply the same type of loading along both axes (both axes in tension, or both axes in compression), as well as mixed mode loading, *i.e.* one axis can apply tension and the other axis can apply compression simultaneously. To stop the motors, the user can enter “0” (zero) into the serial monitor.

Second program: load cell calibration

Before the initial use of the load cell, it must be calibrated. This can be done once, and repeated if load cell readings are showing unrealistic or wrong data. The load cell used to develop and test the calibration program had a capacity of 5000 lbf.; however, the procedure is mostly the same for any load cell capacity. The only difference would be the voltage requirements since the Arduino Uno microcontroller can only supply up to 5 V. For high capacity load cells, as is the case for this project, an external power source capable of providing the required voltage must be used. A calibration factor is used to convert the change in voltage in the load cell when a load is applied, to units of mass. In the case of the 5000 lbf. capacity load cell, the calibration factor was -650. After uploading the program to the Arduino Uno, there should be no load applied to the load cell, because the output is first tared. Following the tare, a baseline reading is obtained, which is used as the “zero factor’; this is a value that the load cell outputs when there is no load applied on to it, and it can be used in place of taring by subtracting this value from the output of the load cell. The calibration code is available for reference in Appendix D.

Third program: reading and displaying load cell output

After calibrating the load cell, the calibration factor is entered into the Arduino program that is used for reading and displaying the load cell output. This can be seen on line 26 of the program in Appendix E. When this program is uploaded to the Arduino Uno, it first tares the load cell, so there should be no load initially on the load cell. This small program will then display the load cell output onto the serial monitor. It's worth noting again that the units of the load cell output are in pounds (lbf.) for the calibration factor of -650. To change the units, the calibration program must be run again in order to find the new calibration factor for the desired units.

At the current stage, the software for controlling the stepper motors is functional, the load cell has been properly calibrated and readings are correct. This is demonstrated by placing masses of known weight on the load cell and the output is equal to the actual weight of the object. However, the physical components of the apparatus are not machined and assembled as of writing this report.

4.1. What Worked

The current code for the stepper motors is capable of taking user inputs for strain rate (rotation speed of stepper motors), loading type - tension or compression (direction of rotation of stepper motors) - through the serial monitor. The Arduino code for the load cells is able to correctly display the load readings on to the serial monitor.

The linear guide rails and bearings lead and ball screws, as well as the steel base plate component are currently available, and their sizes are compatible with each other. These comprise the main components for the final assembly.

4.2. Future Work

Refactoring of the Arduino code to make all software components work together, since the code segment that controls the stepper motors was developed and tested separately from the code that calibrates and displays the readings from the load cell. This was attempted but failed to succeed due to the limitations imposed by the Arduino, including how the Arduino runs code. Furthermore, the libraries used to develop the software may have become out-of-date or deprecated. Refactoring must include updating all libraries and ensuring the current code works correctly with updated libraries.

Furthermore, the software at its current stage may not be very user-friendly as it requires inputs to the serial monitor, as well as the load cell readings displayed on the serial monitor. A simple graphic user interface may be useful to ensure easier operations, and more importantly a function that can record the load cell data and save them on to a text (or .CSV) file for further analysis. Currently the program for reading the load cell output, only displays it on the serial monitor and the values are not recorded. The user can manually copy the output and paste them onto a text file.

A function to halt the stepper motors when specimen failure is detected would also be needed. Specimen failure can be detected by a function that constantly calculates the change in load cell reading, when the reading changes more than a certain percentage (*e.g.* 10%) then the function must send a command to the Arduino that stops the stepper motors.

The focus of future work must be to machine and assemble the arms and grips of the testing apparatus, followed by assembling and connecting to stepper motor and lead screw components. From the arms' components, the items that remain are the L-bracket, U-support, load bearing rod, and grip. Once these components are ready, the assembly of the arms of the apparatus can begin. This is crucial for testing the main function of the apparatus and working out any kinks in the assembly or bugs in the software. After testing the arms, other components may be required in order to attach them to the base plate. Following that, the final wiring and connections to the Arduino can be undertaken.

One safety feature that the apparatus requires is to stop the stepper motors when the arms have moved a certain amount of length, to prevent damage to the motors and other components. It can be a physical switch that activates when the arms reach a certain point along their axis of movement.

APPENDIX A: Demmerle and Boehler - Optimal Design of Biaxial Cruciform Specimens [22]

- Consider symmetrical test area with thickness *2b,*
- Function *f(y, a)* describes shape of test area, where *a* specifies length of area (*e.g.* for a square *f = a*),
- Calculate volume of central test area,

$$V_{tot} = \int_{-b}^{b} \int_{-a}^{a} \int_{-f(y,a)}^{f(y,a)} dx\, dy\, dz$$

- Central test area is continuously subdivided into an infinite number of concentric plates with the same shape as the test area and described by constant thickness of *2b*, function *f(y, p)*, and variable parameter *p*, shown in Figure A1,
- Volume of plates can be calculated,

$$V(p) = \int_{-b}^{b} \int_{-p}^{p} \int_{-f(y,p)}^{f(y,p)} dx\, dy\, dz, \text{ for } 0 \leq p \leq a$$

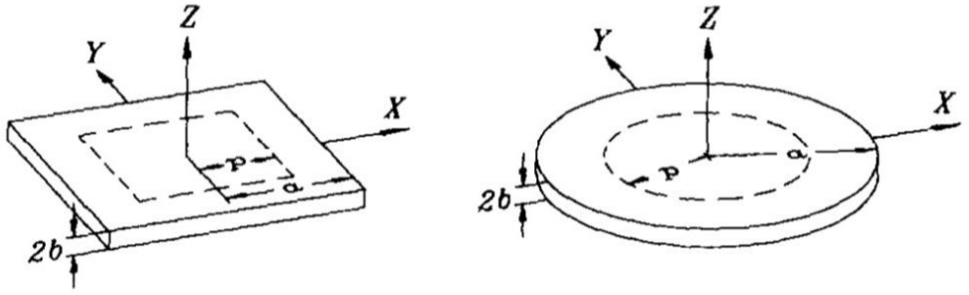


Figure 23 - Illustration of symmetric central test area of biaxial flat specimens with parameters a, b, and p and the function f(y, a). Square (f = a) and circle (f=√(a^2-y^2)) test areas.

- Calculate average σ_{**} and standard deviation s_{**} for stresses σ_{xx}, σ_{xy}, σ_{yy} and Von Mises σ_{vm},

$$\bar{\sigma}_{**}(p) = \frac{1}{V(p)} \int_{-b}^{b} \int_{-p}^{p} \int_{-f(y,p)}^{f(y,p)} \sigma_{**}(x,y,z)\mathrm{d}x\mathrm{d}y\mathrm{d}z$$

$$s_{**}(p) = \left(\frac{1}{V(p)} \int_{-b}^{b} \int_{-p}^{p} \int_{-f(x,p)}^{f(y,p)} \left(\sigma_{**}(x,y,z) - \bar{\sigma}_{**}(p)\right)^2 \mathrm{d}x\mathrm{d}y\mathrm{d}z \right)^{1/2}$$

Where,

$$\sigma_{vm} = \left(\frac{1}{2} \left((\sigma_{xx} - \sigma_{yy})^2 + (\sigma_{yy} - \sigma_{zz})^2 + (\sigma_{zz} - \sigma_{xx})^2 + 6\left(\sigma_{yz}^2 + \sigma_{zx}^2 + \sigma_{xy}^2\right) \right) \right)^{1/2}$$

- Low $s_{**}(p)$values indicate good homogeneity and stress distribution.
- Evaluation of homogeneity of stress distribution,

$$I_1 = \frac{100}{a} \int_0^a \frac{s_{xx}(p)}{\bar{\sigma}_{vm}(p)} \mathrm{d}p$$

$$I_2 = \frac{100}{a} \int_0^a \frac{s_{yy}(p)}{\bar{\sigma}_{vm}(p)} \mathrm{d}p$$

$$I_3 = \frac{100}{a} \int_0^a \frac{s_{xy}(p)}{\bar{\sigma}_{vm}(p)} \mathrm{d}p$$

$$I_4 = \frac{100}{a} \int_0^a \frac{s_{vm}(p)}{\bar{\sigma}_{vm}(p)} \mathrm{d}p$$

- Evaluation of compatibility of stresses $\sigma_{**}(x, y, z)$ in test region with nominal stresses $(\sigma_{**})_n$, where

$$(\sigma_{xx})_n = -\frac{F_x}{(A_x)_{\text{eff}}}$$

$$(\sigma_{yy})_n = \frac{F_y}{(A_y)_{eff}}$$

$$(\sigma_{xy})_n = 0$$

And the applied force in each direction is F_* and its corresponding cross-section area perpendicular to the applied force is $(A_*)_{eff}$.

- Stress compatibility is investigated by using the concentric plates model, with the exception that for each stress component the standard deviation used is $\hat{S}$ with respect to the theoretically determined nominal stress as follows.

$$\hat{s}_{**}(p) = \left(\frac{1}{V(p)} \int_{-b}^{b} \int_{p}^{p} \int_{-f(y,p)}^{f(y,p)} \left(\sigma_{**}(x, y, z) - (\sigma_{**})_n\right)^2 \mathrm{d}x\mathrm{d}y\mathrm{d}z \right)^{1/2}$$

- Normalize the calculated deviation by $\underline{\sigma_{vm}}(p)$ for the considered plate.

$$J_1 = \frac{100}{a} - \int_0^a \frac{\hat{s}_{xx}(p)}{\bar{\sigma}_{vm}(p)} \mathrm{d}p$$

$$J_2 = \frac{100}{a} - \int_0^a \frac{\hat{s}_{yy}(p)}{\bar{\sigma}_{vm}(p)} \mathrm{d}p$$

$$J_3 = \frac{100}{a} \int_0^a \frac{\hat{s}_{xy}(p)}{\tilde{\sigma}_{vm}(p)} \mathrm{d}p$$

- Find weak zones, where initial yielding will occur, characterized by areas with Von Mises stress at their highest level $(\sigma_{vm})_{max}$,
- Initial yielding should occur in central test area where strains are measured,

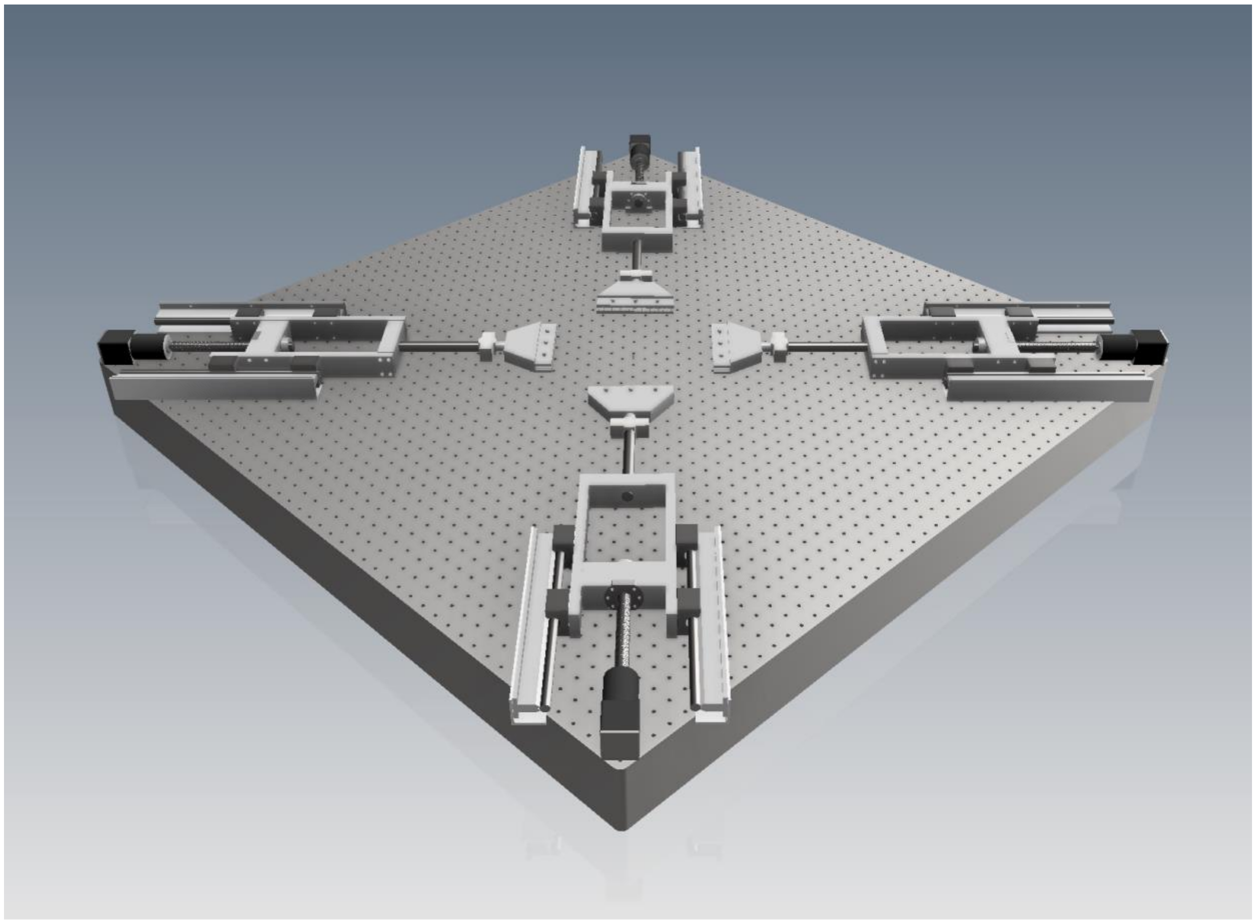
- Component *K* considers this criterion by evaluating difference between $(\sigma_{vm})_{max}$ and $\underline{\sigma}_{vm}(a)$ inside the test section, multiplied by a penalty factor *P*

$$K = \left(\frac{(\sigma_{vm})_{\max}}{\bar{\sigma}_{\nu m}(a)} - 1\right)P$$

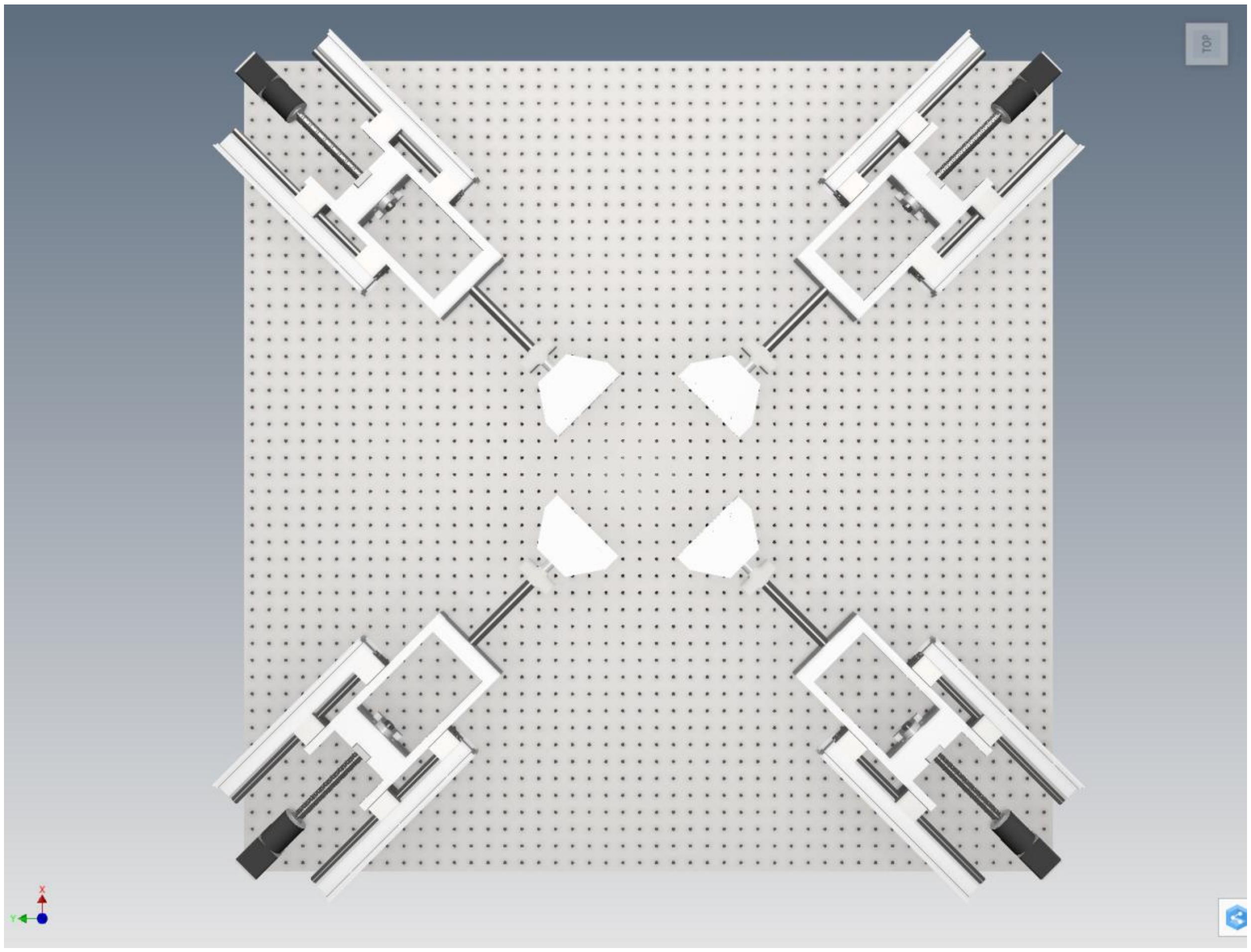
- Depending on the location of maximum Von Mises stress level, the factor *P* can have three different values;
  - *P = 0*; $(\sigma_{vm})_{max}$ located inside plate with length *a/2*, *i.e.* inside the central test section,
  - *P = 20*; $(\sigma_{vm})_{max}$ located between plates, slightly outside of the central test section - penalty is low as initial yielding still occurs inside test section,
  - *P = 100*; $(\sigma_{vm})_{max}$ located outside of the test section, not desirable.

For further details, please refer to the paper by Demmerle and Boehler.

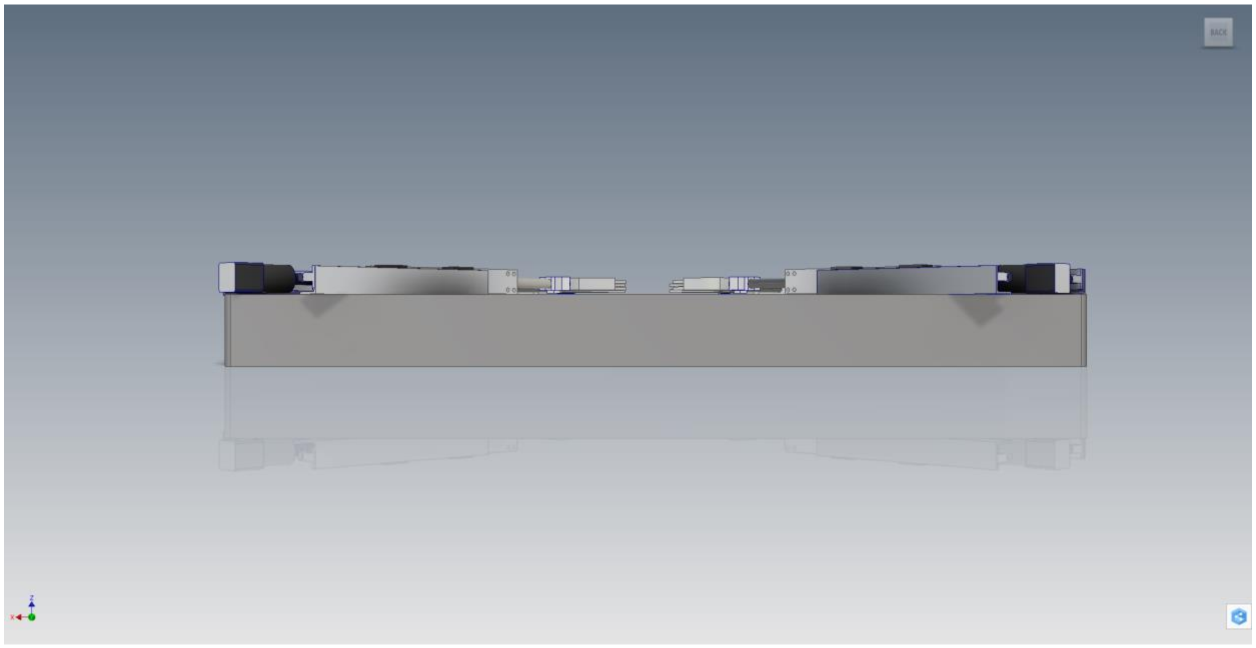
APPENDIX B.1.: 3D CAD Models of Final Assembly of Biaxial Testing Apparatus



Home View

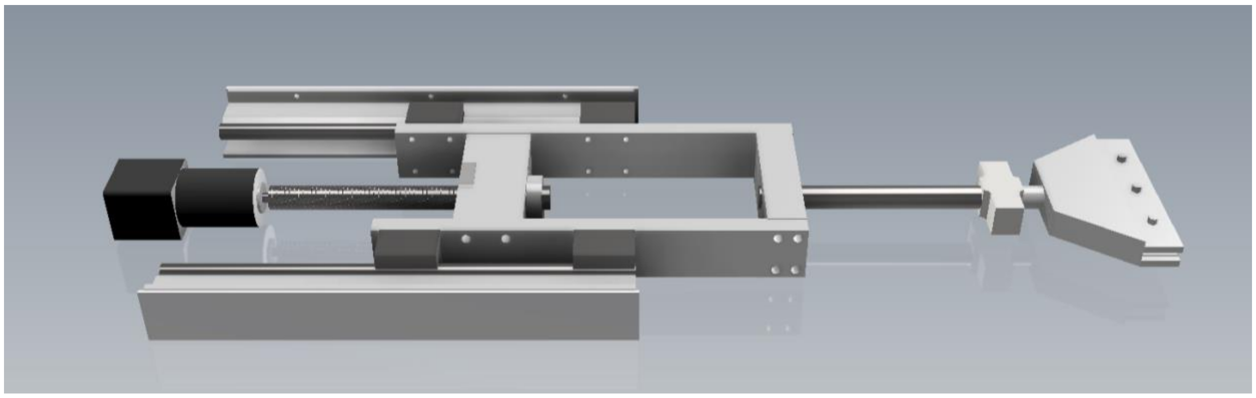


Top View

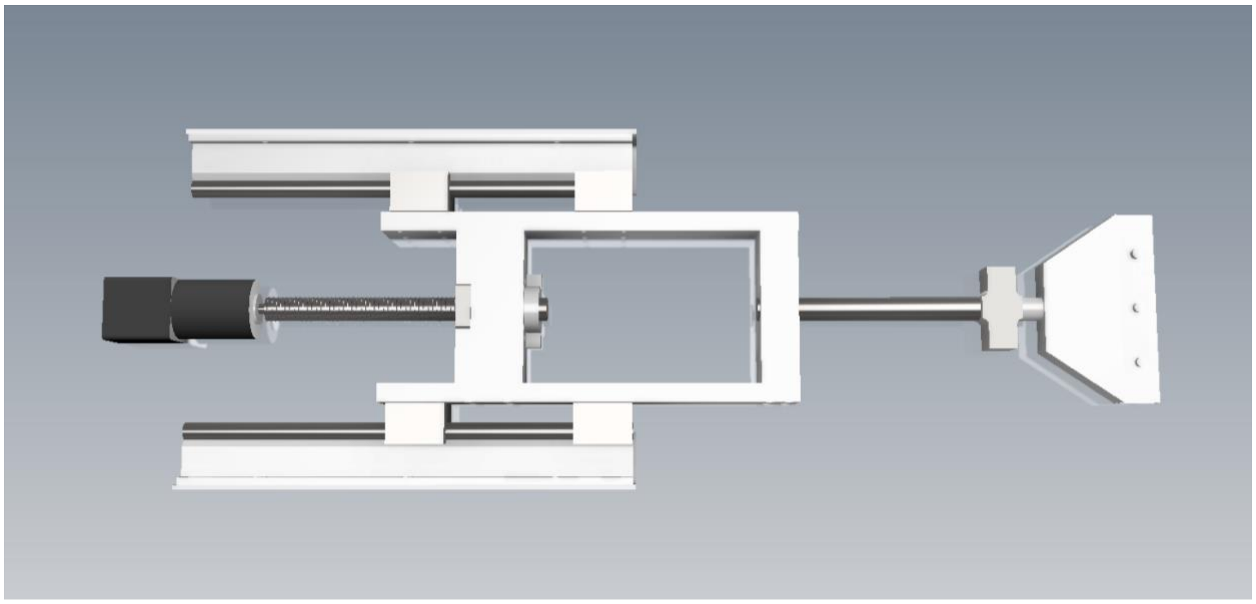


Side View

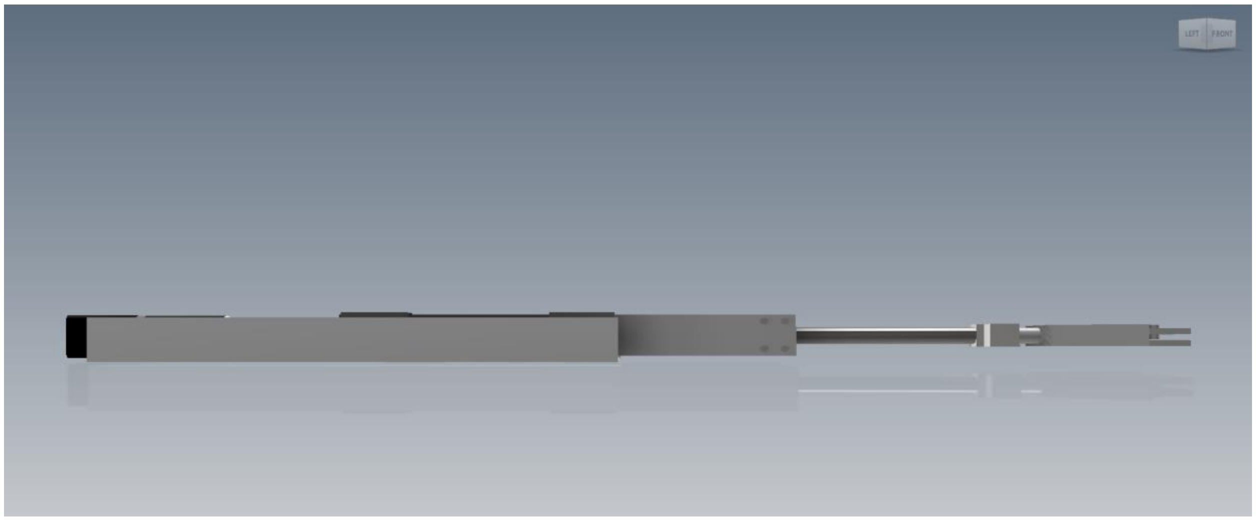
APPENDIX B.2.: 3D CAD Model of Grip and Arm system of Biaxial Testing Apparatus



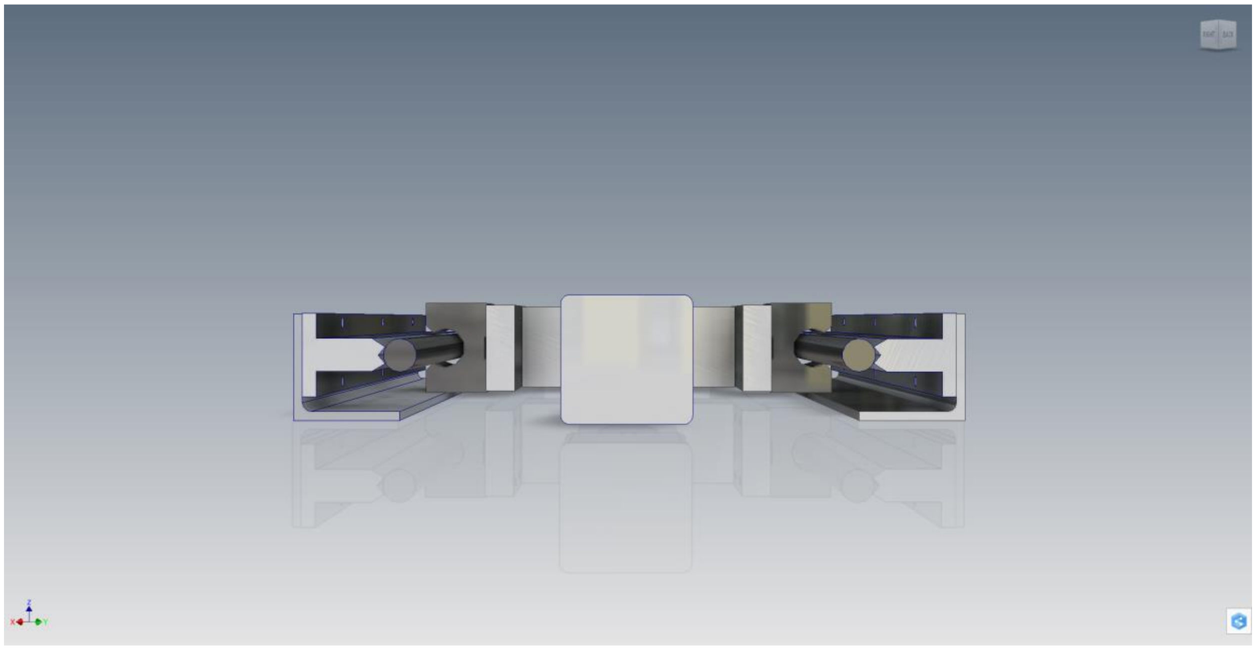
Home View



Top View



Side View



Rear View

APPENDIX C: Arduino program for controlling strain rate and load type (tension/compression)

Note: comments start with // or written between /* and */

Example:

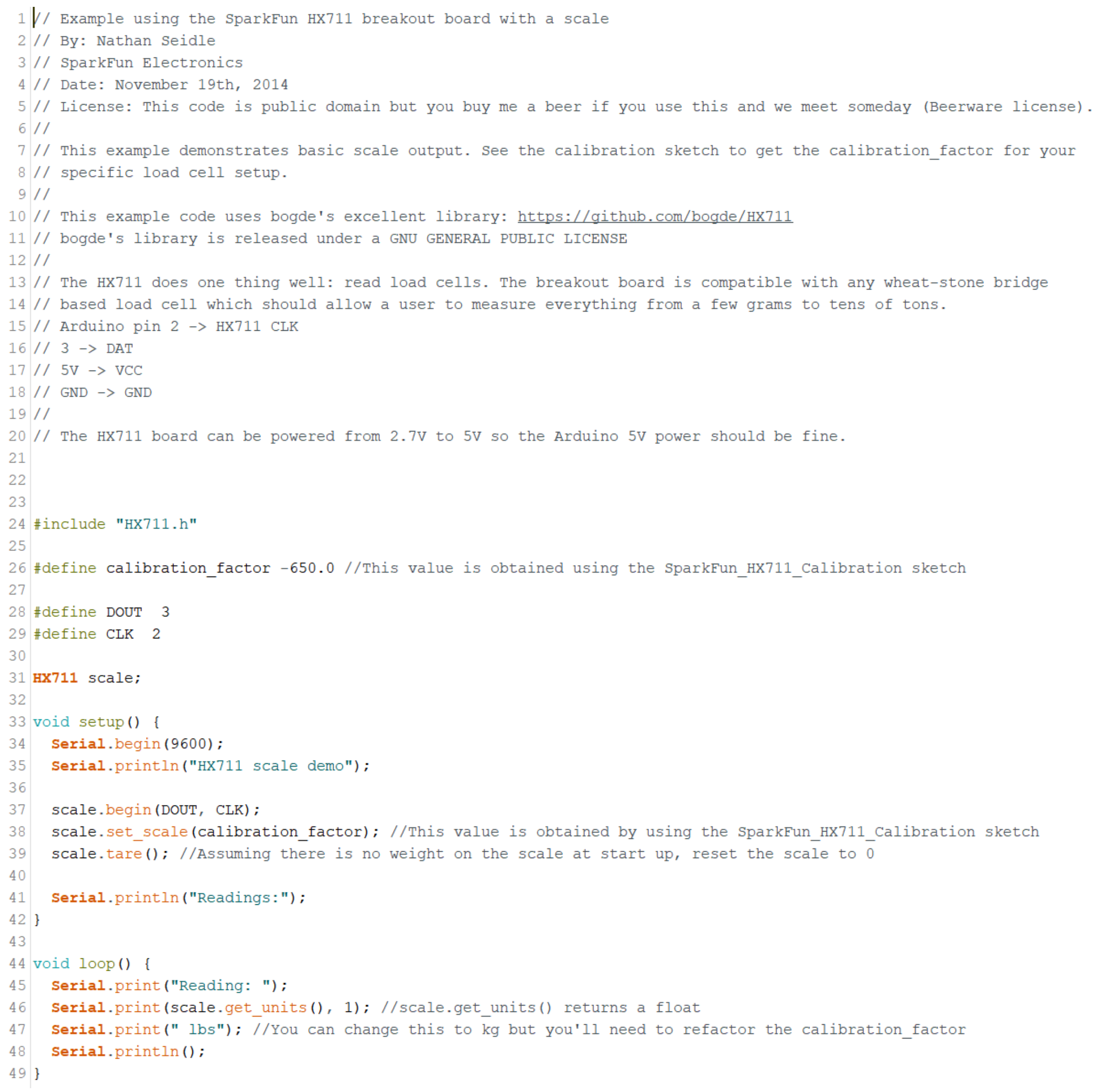
// this is a comment

/* this is a comment */

```
/* Ryerson University - Aerospace Engineering Department
   Project: Biaxial Testing Apparatus
   Created by Amir Noorafkan

   Following code is used for controling the speed and
   direction of rotation of the stepper motors used for
   this project.

   After uploading code, open Serial Monitor to enter values

   May to August 2019

*/


#include <Stepper.h>



const int stepsPerRevolution = 20000;

// stepper function for x-axis motor
Stepper myStepper1(stepsPerRevolution, 3, 4, 5, 6);

// stepper funciton for y-axis motor
Stepper myStepper2(stepsPerRevolution, 10, 11, 12, 13);

```

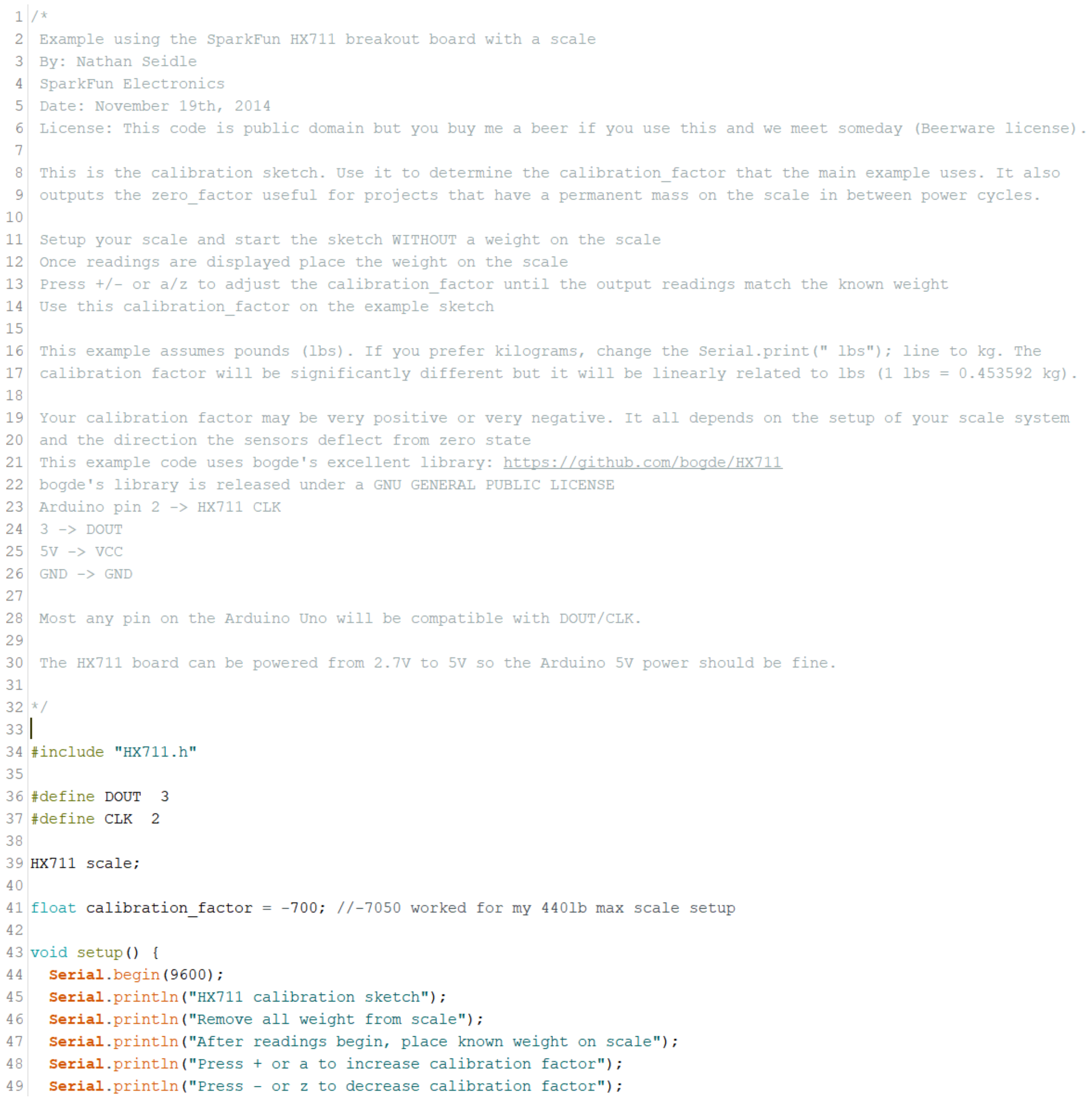
```
// pin setup for x-axis motors
const int pulPin1 = 3;
const int dirPin1 = 4;
const int enaPin1 = 5;
const int optoPin1 = 6;

// pin setup for y-axis motors
const int pulPin2 = 10;
const int dirPin2 = 11;
const int enaPin2 = 12;
const int optoPin2 = 13;

// x-axis motors variables
int stepVar1 = 0;
int dirVar1 = 0;
int strainVar1 = 0;
int enaVar1 = 0;
float strainRate1 = 0.0;

// y-axis motors variables
int stepVar2 = 0;
int dirVar2 = 0;
int strainVar2 = 0;
int enaVar2 = 0;
float strainRate2 = 0.0;

// x-axis
String stepStr1;
String enaStr1;
String dirStr1;
String strainStr1;

// y-axis
String stepStr2;
String enaStr2;
String dirStr2;
String strainStr2;

void setup() {
  Serial.begin(9600);
  pinMode(pulPin1, OUTPUT);
  pinMode(dirPin1, OUTPUT);
  pinMode(enaPin1, INPUT);

  pinMode(pulPin2, OUTPUT);
  pinMode(dirPin2, OUTPUT);
  pinMode(enaPin2, INPUT);

  Serial.print("Select input: ");
  Serial.print("\n");
  Serial.print("\t #1: tension");
  Serial.print("\n");
  Serial.print("\t #2: compression");
  Serial.print("\n");
  Serial.print("\t #0: STOP");
  Serial.print("\n");
  Serial.print("Type strain rate (mm/min), separate with comma.");
  Serial.print("\n");
  Serial.print("Maximum strain rate is 23 mm/min");
  Serial.print("\n");
  Serial.print("e.g. tension test at 7.43 mm/min: 1, 7.43");
}


void loop() {

  while (Serial.available()) {

    dirStr1 = Serial.readStringUntil(',');
    Serial.read();
    strainStr1 = Serial.readStringUntil(',');
    Serial.read();

    dirVar1 = dirStr1.toInt();
    strainRate1 = strainStr1.toFloat();
    strainVar1 = strainRate1 / (0.3037);

    dirStr2 = Serial.readStringUntil(',');
    Serial.read();
    strainStr2 = Serial.readStringUntil(',');
    Serial.read();

    dirVar2 = dirStr2.toInt();
    strainRate2 = strainStr2.toFloat();
    strainVar2 = strainRate2;
    // / (0.3037);

    Serial.print("\n");
    Serial.print("\n");
    Serial.print("Test #");
    Serial.print(dirVar1);
    Serial.print(" started");
    Serial.print("\n");
    Serial.print("Strain rate set to: ");
    Serial.print(strainRate1);
    Serial.print(" mm/min");

    Serial.print("\n");
    Serial.print("\n");
    Serial.print("Test #");
    Serial.print(dirVar2);
    Serial.print(" started");
    Serial.print("\n");
    Serial.print("Strain rate set to: ");
    Serial.print(strainRate2);
    Serial.print(" mm/min");
  }

  if ((dirVar1 == 1) && (dirVar2 == 1)) {
    // X and Y rotate clockwise (tension both axis)
    myStepper1.step(-stepsPerRevolution);
//    myStepper2.step(-100);
    myStepper1.setSpeed(strainVar1);
//    myStepper2.setSpeed(strainVar2);
  }

  if ((dirVar1 == 2) && (dirVar2 == 2)) {
    // X and Y rotate counterclockwise (compression both axis)
    myStepper1.step(100); //low value to
    myStepper2.step(100);
    myStepper1.setSpeed(strainVar1);
    myStepper2.setSpeed(strainVar2);
  }

  if ((dirVar1 == 1) && (dirVar2 == 2)) {
    // X rotate counterclockwise, Y rotate clockwise (tension in X and compression in Y)
    myStepper1.step(100);
    myStepper2.step(-100);
    myStepper1.setSpeed(strainVar1);
    myStepper2.setSpeed(strainVar2);
  }

  if ((dirVar1 == 2) && (dirVar2 == 1)) {
    // X rotate counterclockwise, Y rotate clockwise,  (compression in X and tension in Y)
    myStepper1.step(-100);
    myStepper2.step(100);
    myStepper1.setSpeed(strainVar1);
    myStepper2.setSpeed(strainVar2);
  }

  if ((dirVar1 == 0) && (dirVar2 == 1)) {
    // X not moving, Y rotate clockwise (only tension in Y)
    myStepper1.step(0);
    myStepper2.step(100);
    myStepper1.setSpeed(0);
    myStepper2.setSpeed(strainVar2);
  }

  if ((dirVar1 == 0) && (dirVar2 == 2)) {
    // X not moving, Y rotate counterclockwise (only compression in X)
    myStepper1.step(0);
    myStepper2.step(-100);
    myStepper1.setSpeed(0);
    myStepper2.setSpeed(strainVar2);
  }

  if ((dirVar1 == 1) && (dirVar2 == 0)) {
    // X rotate clockwise, Y not movine (only tension in X)
    myStepper1.step(100);
    myStepper2.step(0);
    myStepper1.setSpeed(strainVar1);
    myStepper2.setSpeed(0);
  }

  if ((dirVar1 == 2) && (dirVar2 == 0)) {
    // X rotate counterclockwise, Y not moving (only compression in X)
    myStepper1.step(-100);
    myStepper2.step(0);
    myStepper1.setSpeed(strainVar1);
    myStepper2.setSpeed(0);
  }

  if (dirVar1 == 0) {
    // stop:
    myStepper1.step(0);
    myStepper2.step(0);

    // shut off power
    PORTD = B00000000;
    PORTB = B00000000;
  }
}
```

APPENDIX D: Arduino program for displaying calibrated load cell values on to the serial monitor

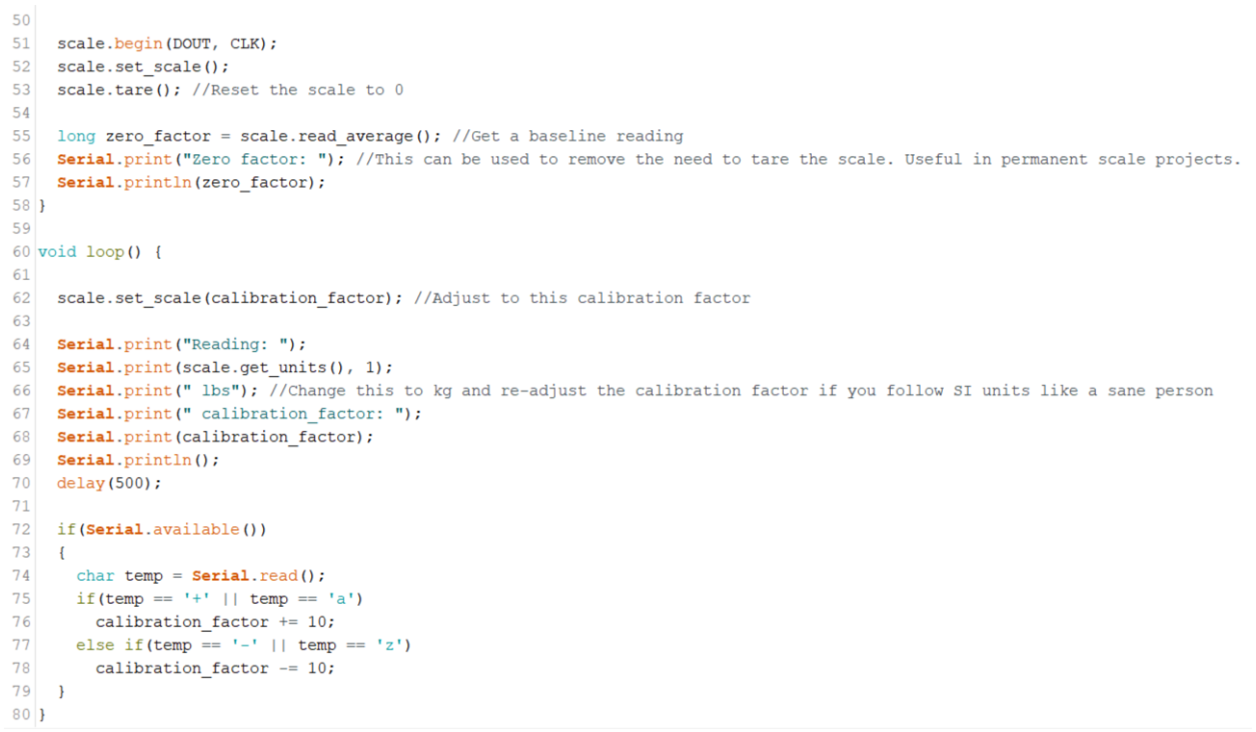


```
// Example using the SparkFun HX711 breakout board with a scale
// By: Nathan Seidle
// SparkFun Electronics
// Date: November 19th, 2014
// License: This code is public domain but you buy me a beer if you use this and we meet someday (Beerware license).
//
// This example demonstrates basic scale output. See the calibration sketch to get the calibration_factor for your
// specific load cell setup.
//
// This example code uses bogde's excellent library: https://github.com/bogde/HX711
// bogde's library is released under a GNU GENERAL PUBLIC LICENSE
//
// The HX711 does one thing well: read load cells. The breakout board is compatible with any wheat-stone bridge
// based load cell which should allow a user to measure everything from a few grams to tens of tons.
// Arduino pin 2 -> HX711 CLK
// 3 -> DAT
// 5V -> VCC
// GND -> GND
//
// The HX711 board can be powered from 2.7V to 5V so the Arduino 5V power should be fine.



#include "HX711.h"

#define calibration_factor -650.0 //This value is obtained using the SparkFun_HX711_Calibration sketch

#define DOUT  3
#define CLK  2

HX711 scale;

void setup() {
  Serial.begin(9600);
  Serial.println("HX711 scale demo");

  scale.begin(DOUT, CLK);
  scale.set_scale(calibration_factor); //This value is obtained by using the SparkFun_HX711_Calibration sketch
  scale.tare(); //Assuming there is no weight on the scale at start up, reset the scale to 0

  Serial.println("Readings:");
}

void loop() {
  Serial.print("Reading: ");
  Serial.print(scale.get_units(), 1); //scale.get_units() returns a float
  Serial.print(" lbs"); //You can change this to kg but you'll need to refactor the calibration_factor
  Serial.println();
}
```

APPENDIX E: Arduino program for reading and displaying load cell data



```
/*
 Example using the SparkFun HX711 breakout board with a scale
 By: Nathan Seidle
 SparkFun Electronics
 Date: November 19th, 2014
 License: This code is public domain but you buy me a beer if you use this and we meet someday (Beerware license).

 This is the calibration sketch. Use it to determine the calibration_factor that the main example uses. It also
 outputs the zero_factor useful for projects that have a permanent mass on the scale in between power cycles.

 Setup your scale and start the sketch WITHOUT a weight on the scale
 Once readings are displayed place the weight on the scale
 Press +/- or a/z to adjust the calibration_factor until the output readings match the known weight
 Use this calibration_factor on the example sketch

 This example assumes pounds (lbs). If you prefer kilograms, change the Serial.print(" lbs"); line to kg. The
 calibration factor will be significantly different but it will be linearly related to lbs (1 lbs = 0.453592 kg).

 Your calibration factor may be very positive or very negative. It all depends on the setup of your scale system
 and the direction the sensors deflect from zero state
 This example code uses bogde's excellent library: https://github.com/bogde/HX711
 bogde's library is released under a GNU GENERAL PUBLIC LICENSE
 Arduino pin 2 -> HX711 CLK
 3 -> DOUT
 5V -> VCC
 GND -> GND

 Most any pin on the Arduino Uno will be compatible with DOUT/CLK.

 The HX711 board can be powered from 2.7V to 5V so the Arduino 5V power should be fine.

*/

#include "HX711.h"

#define DOUT  3
#define CLK  2

HX711 scale;

float calibration_factor = -700; //-7050 worked for my 440lb max scale setup

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

  long zero_factor = scale.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);
}

void loop() {

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

  Serial.print("Reading: ");
  Serial.print(scale.get_units(), 1);
  Serial.print(" lbs"); //Change this to kg and re-adjust the calibration factor if you follow SI units like a sane person
  Serial.print(" calibration_factor: ");
  Serial.print(calibration_factor);
  Serial.println();
  delay(500);

  if(Serial.available())
  {
    char temp = Serial.read();
    if(temp == '+' || temp == 'a')
      calibration_factor += 10;
    else if(temp == '-' || temp == 'z')
      calibration_factor -= 10;
  }
}
```

REFERENCES

[1] “A350 XWB Family,” *Airbus*. [Online]. Available: https://www.airbus.com/aircraft/passenger-aircraft/a350xwb-family.html. [Accessed: 04-Feb-2020].

[2] A. Makinde, L. Thibodeau, and K. W. Neale, “Development of an apparatus for biaxial testing using cruciform specimens,” *Experimental Mechanics*, vol. 32, no. 2, pp. 138–144, Jun. 1992, doi: 10.1007/BF02324725.

[3] K. Morris, A. Rosenkranz, H. Seibert, L. Ringel, S. Diebels, and F. E. Talke, “Uniaxial and biaxial testing of 3D printed hyperelastic photopolymers,” *Journal of Applied Polymer Science*, vol. 137, no. 8, p. 48400, 2020, doi: 10.1002/app.48400.

[4] “Mooney-Rivlin.” [Online]. Available: https://www.continuummechanics.org/mooneyrivlin.html. [Accessed: 06-Feb-2020].

[5] Douglas P. Holmes, “Mechanics of Materials: Stress,” *Boston University Mechanical Engineering*. [Online]. Available: https://www.bu.edu/moss/mechanics-of-materials-stress/. [Accessed: 06-Feb-2020].

[6] M. M. Team, “Types of Loads,” *ME Mechanical*, 07-Jun-2016. [Online]. Available: https://me-mechanicalengineering.com/types-of-loads/. [Accessed: 06-Feb-2020].

[7] M. D. Hayes, D. B. Edwards, and A. R. Shah, “4 - Fractography Basics,” in *Fractography in Failure Analysis of Polymers*, M. D. Hayes, D. B. Edwards, and A. R. Shah, Eds. Oxford: William Andrew Publishing, 2015, pp. 48–92.

[8] J. P. Boehler, S. Demmerle, and S. Koss, “A new direct biaxial testing machine for anisotropic materials,” *Experimental Mechanics*, vol. 34, no. 1, pp. 1–9, Mar. 1994, doi: 10.1007/BF02328435.

[9] “Stress in Thin-Walled Tubes or Cylinders.” [Online]. Available: https://www.engineeringtoolbox.com/stress-thin-walled-tube-d_948.html. [Accessed: 07-Feb-2020].

[10] P. Beccarelli, B. Martin, B. Lau, J. Chilton, S. Bonardi, and R. Maffei, “Single-board microcontrollers applied to biaxial tests for architectural membranes,” *International Association for Shell and Spatial Structures (IASS)*, p. 9, 2015.

[11] “Stress (mechanics),” *Wikipedia*. 04-Feb-2020.

[12] “Shear stress | physics,” *Encyclopedia Britannica*. [Online]. Available: https://www.britannica.com/science/shear-stress. [Accessed: 09-Feb-2020].

[13] Krishnavedala, *Shear stress simple image*. 2012.

[14] H.-C. Wu, *Continuum mechanics and plasticity*. Boca Raton: Chapman & Hall/CRC, 2005.

[15] Douglas P. Holmes, “Mechanics of Materials: Strain,” *Boston University Mechanical Engineering*. [Online]. Available: https://www.bu.edu/moss/mechanics-of-materials-strain/. [Accessed: 09-Feb-2020].

[16] Pankaj Mishra, “Stress Strain Curve – Relationship, Diagram and Explanation - Mechanical Booster,” *Mechanical Booster*, 06-Sep-2016. [Online]. Available: https://www.mechanicalbooster.com/2016/09/stress-strain-curve-relationship-diagram-explanation.html. [Accessed: 09-Feb-2020].

[17] PAR, Diagram of an isotropic linear elastic material with Poisson ratio of 0.5 subject to axial forces along the x axis only. 2010.

[18] “Planar Biaxial Test Machines,” *TestResources Inc.* [Online]. Available: https://www.testresources.net/test-machines/planar-biaxial-test-machines/. [Accessed: 05-Feb-2020].

[19] “Biaxial and Triaxial Testing of Biomaterials.” [Online]. Available: https://www.zwickroell.com/en/medical/biomaterials-clinical-research/biaxial-triaxial-tests. [Accessed: 05-Feb-2020].

[20] “Planar biaxial testing system | Biomedical Engineering.” [Online]. Available: https://cse.umn.edu/bme/planar-biaxial-testing-system. [Accessed: 05-Feb-2020].

[21] P. Beccarelli, *Biaxial Testing for Fabrics and Foils*. Cham, Switzerland: Springer, 2015.

[22] S. Demmerle and J. P. Boehler, “Optimal design of biaxial tensile cruciform specimens,” *Journal of the Mechanics and Physics of Solids*, vol. 41, no. 1, pp. 143–181, Jan. 1993, doi: 10.1016/0022-5096(93)90067-P.

[23] D. S. Dawicke and D. Pollock, “Biaxial Testing of 2219-T87 Aluminum Using Cruciform Specimens,” National Aeronautics and Space Administration, Langley Research Center, Hampton, Virginia 23681-0001, 4782, Aug. 1997.

[24] “ISO 16842:2014 - Metallic materials - Sheet and strip - Biaxial tensile testing method using a cruciform test piece.” [Online]. Available: https://webstore.ansi.org/Standards/ISO/ISO168422014. [Accessed: 16-Feb-2020].

[25] “Ball Screws - Steel Ball Recirculation | MISUMI Blog.” [Online]. Available: https://blog.misumiusa.com/ball-screw-steel-ball-recirculation/. [Accessed: 05-Mar-2020].

[26] bogde, *bogde/HX711*. 2020, URL: https://github.com/bogde/HX711.

[27] N. E. Dowling, Mechanical Behavior of Materials: Engineering Methods for Deformation, Fracture, and Fatigue, 3rd ed. Pearson Prentice Hall, 2007.