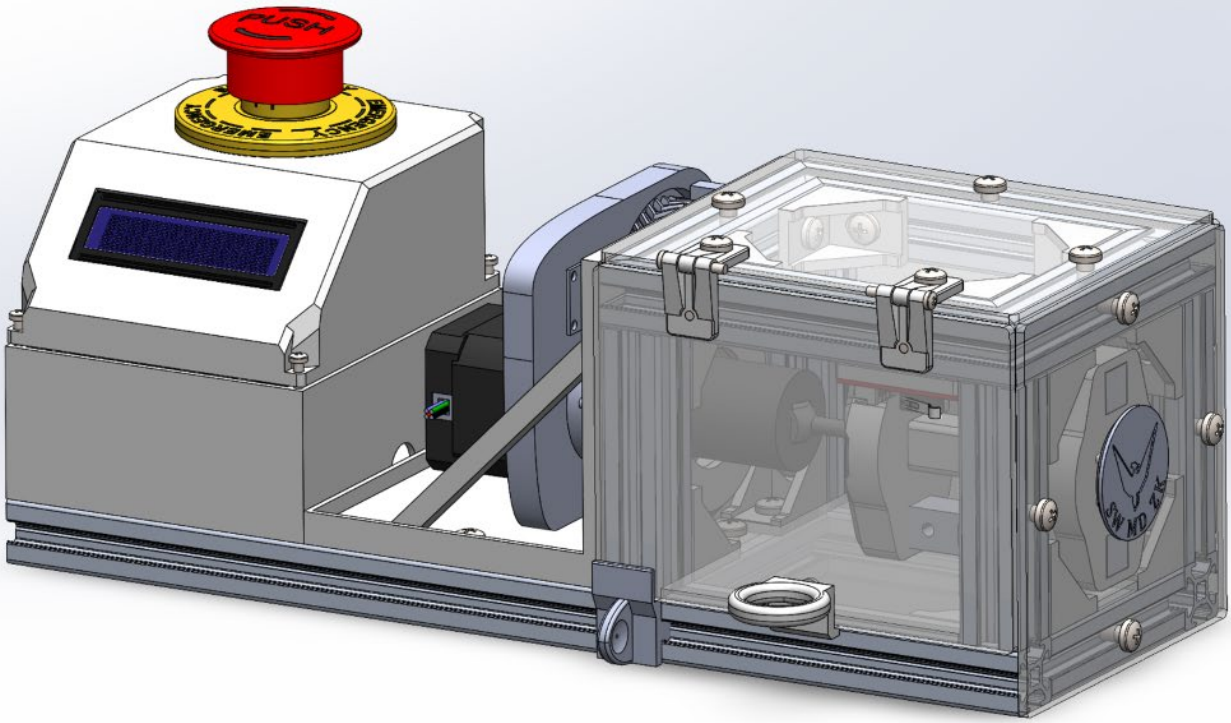


SMZ Series 1.0
Torsion Test Device

User Guide

Sydney Wickett, Matthew DeCicco, Zachary Kaiser

⚠ WARNING

Read and fully understand operator's manual before using this device.

Do NOT operate this device without securing the gearbox cover.

Failure to adhere may result in serious injury.

Table of Contents

SYSTEM OVERVIEW

DESCRIPTION

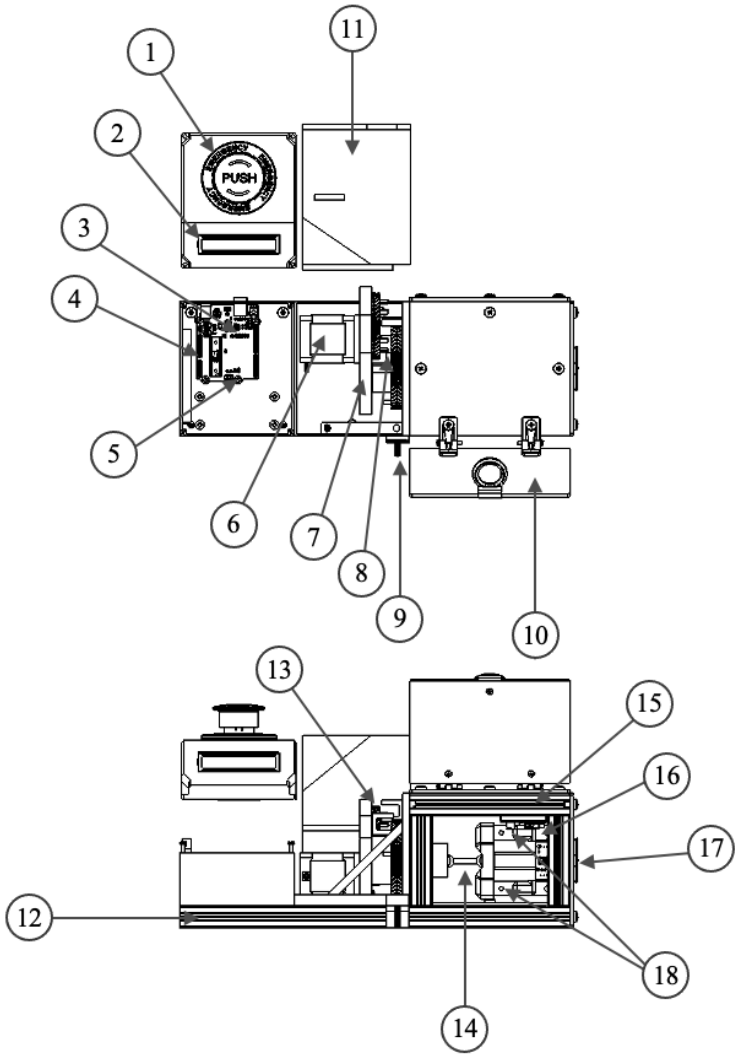
The SMZ Series 1.0 Torsion Tester is a compact, table-top device used to assess the mechanical properties of a material associated with its response to an applied twisting moment, namely its shear modulus and ultimate shear stress. These quantities are indicative of a material's ability to withstand a torsional loading pattern. The device consists of three subsystems:

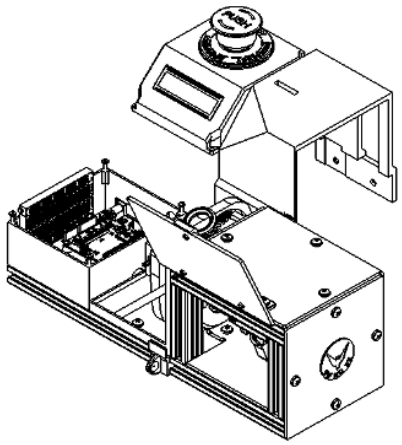
- Structure
- Electronics and Controls
- Mechanical System

The Structure sub-system utilizes 3D-printed compartments and an acrylic enclosure to house the Electronics and Controls and Mechanical System sub-systems and provides support to the device with an aluminum extrusion frame. Stability to the frame is added with four feet, placed on each of the four corners. The device is also easily transportable with a nylon handle.

The Electronics and Controls sub-system consists of the Arduino Uno microcontroller board to which the Arduino code is uploaded. The Arduino code facilitates the operation of the motor, collection of data, and data display on the liquid-crystal display (LCD). It is housed by a 3D-printed compartment. The Arduino code needed to operate the device has already been uploaded to the microcontroller, thus it is not needed by the user. MATLAB is the only software needed to operate the device, specifically for data analysis purposes. The MATLAB code, provided in the Appendix, will automatically generate a torque vs. angle of twist plot, a shear stress vs. shear strain plot, and a window containing percent error values for the measured shear modulus and ultimate shear stress values with consideration of literature-based values. It will also notify the user if a load cell error has been detected, an event capable of heavily misconstruing results.

The Mechanical System sub-system encompasses the motor and accompanying gear box (20:1 ratio) and load cell configuration. Between these two components is the location of the sample during a test, slotted into two grips. One grip is fixed, while the other rotates as a result of the motor's rotation. The load cell configuration, positioned at the fixed end, employs two load cells to measure the load associated with the torsional response of the sample, or torque. The angle of twist is taken directly from the motor's angle of displacement. The motor and gear box are enclosed by a gear box cover while the sample and load cell configuration are contained within the acrylic enclosure to protect the user.





Component	Description	Component	Description
1	Emergency Stop Button	10	Testing Enclosure Door
2	LCD	11	Gear Box Encasement
3	Arduino Uno	12	Frame
4	Breadboard	13	Optical Sensors (2)
5	Stepper Driver	14	Sample
6	Stepper Motor	15	Limit Switch
7	Gear Housing	16	Load Cell Amplifier
8	Gears (3)	17	Sample Insertion Site w/ Plunger
9	Latch	18	Load Cells (2)

TECHNICAL SPECIFICATIONS

MOTOR & GEAR BOX		
Motor Type	**-**	Bipolar Stepper
Gear Ratio	**-**	20:1
Torque Capacity	**N-m**	11.8
Twist Angle Capacity	**°**	∞
Rotational Test Speed	**°/min**	36
Angle of Twist Resolution	**°**	± 0.225

LOAD CELLS		
Torque Resolution	**N-m**	0.0025
Display Torque Resolution	**N-m**	0.01
Maximum Load	**kg**	10

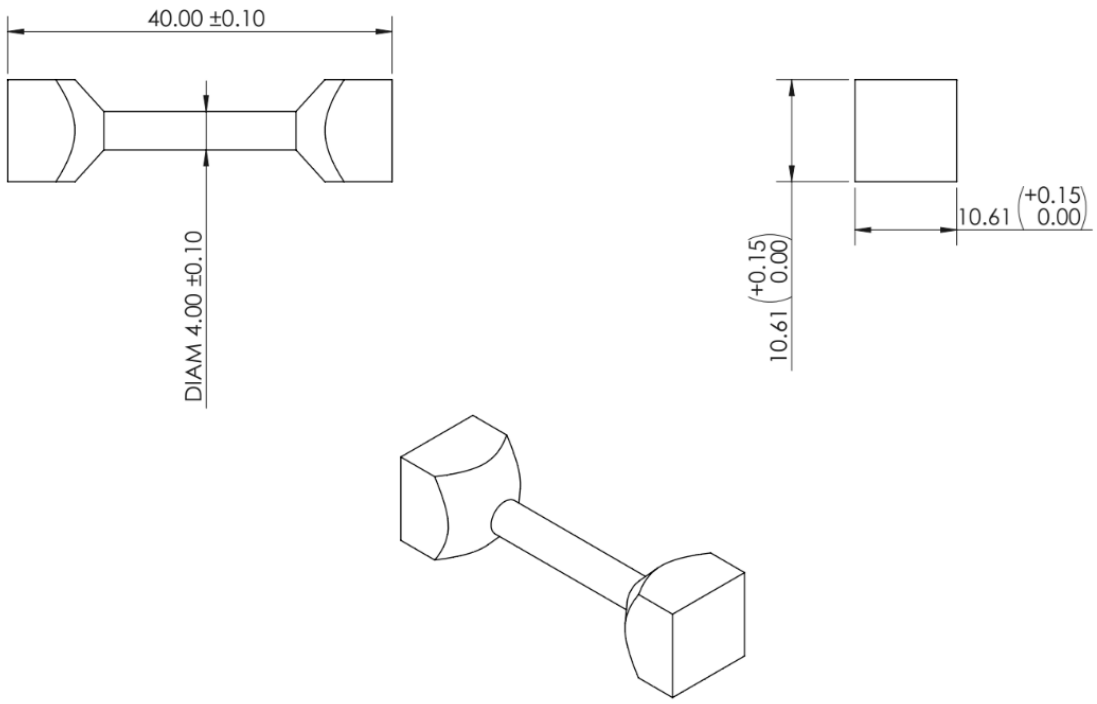
DEVICE		
Mass (Device Only)	**kg**	4

POWER		
Power Requirement	**W**	10

SAMPLE GEOMETRY

The SMZ Series 1.0 Torsion Test Device was designed to test samples with the depicted geometry and does not possess adaptive capabilities for varying geometry.

Note that all units are in mm.



OPERATION

SAMPLE FABRICATION

The standard material of the torsion test device is polylactic acid (PLA). The mechanical properties of PLA are very sensitive to composition and manufacturing processing parameters. The literature values to which the experimental values are based on samples fabricated with the geometry depicted in “Sample Geometry” and using the listed processing parameters.

Processing Parameters for PLA Sample	
PLA Color	Red
Build Orientation	Vertical (90° relative to build plate)
Layer Thickness	0.1 mm
Perimeter	2 Loops
Infill	100%
Nozzle Temperature	230 °C
Fabrication Speed	90 mm/s

CONDUCTING A TORSION TEST

To conduct a torsion test, the following is needed:

- SMZ Series 1.0 Torsion Test Device
- MATLAB
- Personal Computer (PC) (preferably Windows Operating System)
- USB-Arduino Cable
- Power Cable (DC-Outlet)

It may be useful to refer to the diagram on Page 4 throughout the procedure.

For safety purposes, do not operate the device if the gears are uncovered or if the acrylic enclosure is fractured. Additionally, the testing enclosure door must remain closed throughout the entirety of the test and must not be opened unless prompted to do so by the LCD. The emergency stop button may be employed at any time the user deems necessary to end the test. Once the test has begun, do not tamper with the device as this may cause data inconsistencies.

The following procedure is used to conduct a torsion test:

1. The SMZ Series 1.0 Torsion Test Device can be connected to a power supply by plugging a DC cable (akin to a laptop charging cable) into the circular port located on the back of the device and plug the opposite end of the cable into an available outlet.
2. Connect the device to the user's PC using the USB-Arduino cable. Access to the Arduino's port is located on the backside of the device as well.
3. The LCD, indicted on the diagram as Component #2, will then prompt the user to connect via MATLAB. To do so, simply open MATLAB and run the code provided in the Appendix.
4. MATLAB will prompt the user to choose a testing mode. There are two options, real and emulate:
    a. Real: Runs a real test on a sample that has been placed in the testing frame (Type 'R').
    b. Emulate: Runs a simulation based on a previously run test (Type 'E')
        - To choose the file for emulation, type its name in the script as the input for the "importdata" function in Line 44.
5. MATLAB will connect to the Arduino; this is indicated by a message reading "Attempting to connect...".
6. MATLAB will prompt the user to type in the material being tested. This will update the saved file name to contain this name and adjust the literature values for comparison accordingly. This file will also contain the date. Specificity is encouraged when typing the material.

Available Materials:

- Aluminum
- PLA
- ABS

7. The Arduino will be located automatically once the last prompt has been entered.
   - If using a PC with an iOS operating system, the Arduino will need manual connection. Identify the port to which the Arduino is connected by opening Arduino IDE, then copy and paste the port name into the code where it is stated “Replace ‘COM23’ with your location” (Line 38) Close Arduino IDE once complete.
8. The LCD will confirm that the device has successfully connected to MATLAB, and the device will begin to home itself (reset the motors to starting position).
9. The LCD will prompt the user to load the sample. To do so, take the sample and place it into the slot indicted by #17. Push the sample through using the plunger, then open the door to the testing enclosure by sliding the latch to the left. The sample can then be slotted into place by moving the suspended end into the empty square cavity. Once the sample has been securely placed in the grips, close the door and replace the latch by sliding it to the right over the door.
10. A new prompt will appear in MATLAB. To start the test, type ‘Start’. The load cells will zero themselves, and the test will begin.
11. As the test progresses, the torque vs. angle of twist plot will generate in real-time. The torque and angle of twist at any given moment in the test can also be ascertained by looking at the live plot or at the LCD.
12. Once the sample breaks, the test will end. The test rate will be displayed on the LCD. The applied offset will also be displayed in the Command Window on MATLAB. The device will perform a “clean up” and ensuring the sample has completely broken by twisitng the motor an additional 90°.
13. The broken sample can be retrieved by opening the testing enclosure door and removing it.

DATA ANALYSIS

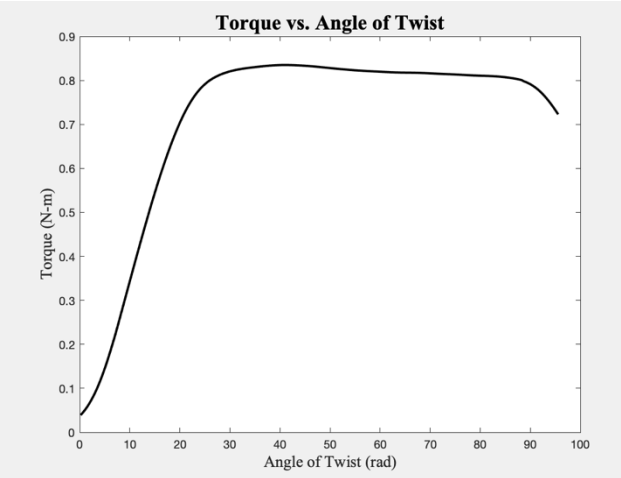
Data analysis is performed for the user by MATLAB. The complete torque vs. angle of twist plot for the test will be displayed.

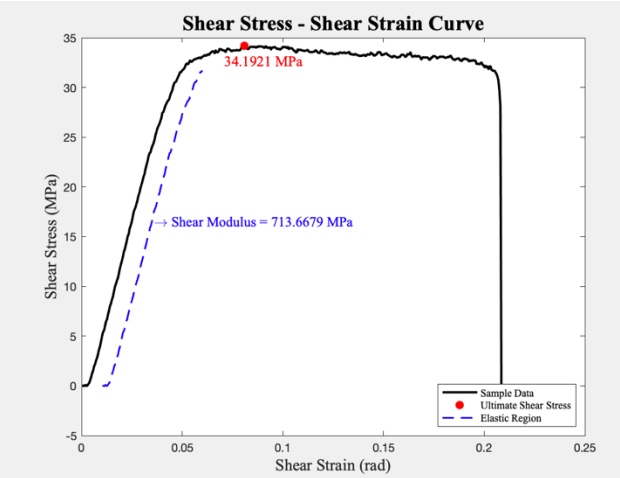
The shear stress vs. shear strain curve for the sample will be automatically generated post-test and displayed to the user. This graph also displays the experimental shear modulus and ultimate shear stress.

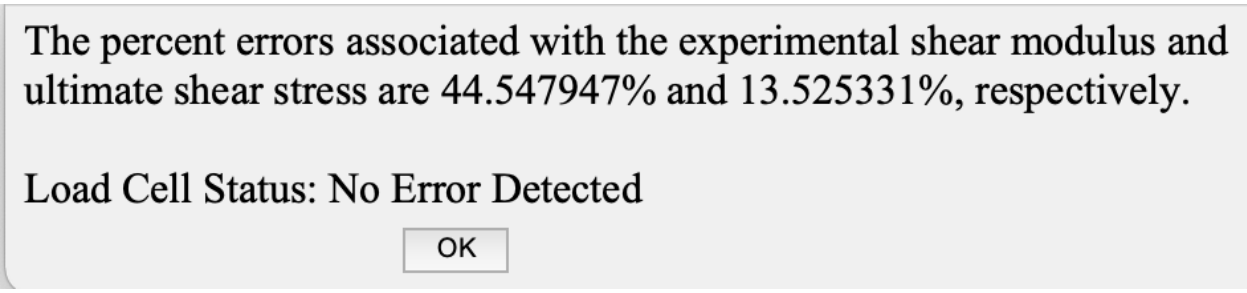
A third pop-up will state the percent errors associated with the experimental values after comparing them to literature values. Note that a change in material not listed under available materials will require a change in these literature values, which can be altered in Lines 194-196 of the MATLAB script for shear modulus and ultimate shear stress.

The third pop-up will also notify the user if a load cell error has been detected.

Samples of the pop-ups are provided.







TROUBLESHOOTING

POTENTIAL PROBLEMS AND THEIR SOLUTIONS

1. **Problem: MATLAB is unable to connect to the Arduino Uno.**
   Solution: Ensure that Arduino IDE is currently open as it will block MATLAB from connecting to the Arduino. Alternatively, if you are using an iOS operating system, the Arduino will need to be manually connected as outlined in Step 6 of the procedure.

2. **Problem: The test ended but the sample is not completely broken and has become stuck in the slots.**
   Solution: This means the sample may have fractured, but not completely broken. The sample can be cut in half with sharp scissors or clippers, **carefully**. Try to make the cut where the sample appears weak to be safer.

3. **Problem: The gears are not moving properly.**
   Solution: Remove the gear box cover and inspect the gears. It is possible that they have loosened, causing irregular movement. If this does not appear to be the case, it is possible that they may need replacement.
   Note: Do NOT apply lubricant to the gears.

4. **Problem: There was a load cell error detected.**
   Solution: Load cell errors can be troublesome for acquiring accurate data, and the severity of the load cell error varies from case to case. If the load cell has clearly interfered with the accuracy of the data, it is best to run another test.

5. **Problem: The percent error values were unexpectedly high, but there was no load cell error detected.**
   Solution: Ensure that the literature values being used for comparison correspond to the material being tested. It is also possible that the sample, specifically PLA, is simply weaker or stronger than normal due to small variations in the printing process that may have occurred. It is best to run another test to confirm the results.

6. **Problem: The sample has become lodged in the insert tube and is stuck.**
   Solution: This issue will occur when a sample has not been properly prepared and trimmed to remove excess filament. To dislodge the sample, depending on where in the tube the sample is located, it can be pushed through with fingers (if located closer to the exit) or pushed through with substantial force by the plunger (if located closer to the entrance).

APPENDIX

MATLAB Code

Copy and paste the following code into MATLAB for device operation:

```
clc, close all, clear all
open = 0;
warning('off','serialport:serialport:ReadlineWarning')
prompt = "Emulate or Real test('E' or 'R'): ";
type = input(prompt,"s");
if type == "E"
    emulate = 1;
else
    emulate = 0;
end
%% Auto. finds the arduino and connects to it
if ~emulate
    fprintf("\nAttempting to connect...(Approx 30 sec)\n")
    dev_name = 'Arduino Uno';
    [~,res]=system('wmic path Win32_SerialPort');
    ind = strfind(res,dev_name);
    if (~isempty(ind))
        port_name = res(ind(1)+length(dev_name)+2:ind(1)+length(dev_name)+6);
        port_name = strip(port_name);
        port_name = strip(port_name,'right',')'); %handling for the port
being 2 digits
        fprintf('%s found on COM-port: %s\n',dev_name, port_name);

        try
            arduino = serialport(port_name,115200); %%The arduino object at
115200 baud
            fprintf('%s is opened\n',port_name);
            open = 1;
        catch err
            fprintf('%s\n%s\n',err.identifier,err.message);
        end
    else
        fprintf('COM-port is not found\nPlug in device, manually enter COM
Port, or Emulate the test\n');
        open = 0;
    end
end

%% Manually connects to Arduino
if open == 0 && ~emulate
    arduino = serialport("COM23",115200); %Replace 'COM23' with your location
    open = 1;
    fprintf("Manual Opening Successful\n");
end
%% emualte import
if emulate
    dataE = importdata("TwistAngleTorqueSampleData_RED_13.61E_Apr-21-
2023.mat","dataNice");
end
%% Data collection
```

```
if open || emulate
    loadCellE = 0;
    if ~emulate
        arduino.Timeout = 10;
    end
    angle = []; %angle vector
    torque =[]; %torque vector
    dataV = [];
    a1=[];
    ready = 0;
    if emulate
        materialIn = "\nWhat is the material: ";
        material = input(materialIn,"s");
        ready = 1;
    else
        materialIn = "\nWhat is the material: ";
        material = input(materialIn,"s");
        fprintf("-=-=-=-=-=-=-=-=-=-=-=-\nFollow prompts on LCD\n");
    end

    while ~ready
        writeline(arduino,"MATLAB");
        data = readline(arduino);
        if ~isempty(data)
            data = strip(data);
            if data == "ready"
                ready = 1;
            end
        end
    end
    done = 0;
    if ~emulate
        prompt = "Enter command (start or home): ";
        command = input(prompt,"s");
        writeline(arduino,command);
    end
    livePlot = animatedline;
    title("Torque vs. Angle of Twist", 'FontSize', 18, 'FontName', 'Times')
    xlabel("Angle of Twist (deg)", 'FontSize', 14, 'FontName', 'Times')
    ylabel("Torque (N-m)", 'FontSize', 14, 'FontName', 'Times')
    while ~done
        data = "";
        if ~emulate
            data = readline(arduino);
            if isempty(data)
                data = "";
                if length(dataV)>1
                    done = 1;
                end
            end
        end

        if strlength(data) > 0 && ~done && ~emulate
            data = strip(data);
            if data == "EC"
                fprintf("\nCAPACITY EXCEEDED!\n");
```

```
                done = 1;
            elseif data == "LC"
                loadCellE = 1;
                fprintf("\n*Warning: Load Cell Error Detected*\n");
            elseif data == "B"
                done = 1;
            else
                dataV = [dataV;data];
                newStr = split(data,':');
                angle = [angle;str2double(newStr(1))];
                torque = [torque;str2double(newStr(2))];
                xlim([0 max(angle)+5]);
                ylim([0 max(torque)+.2]);
                addpoints(livePlot,angle(end),torque(end));
                dim = [0.15 0.6 0.3 0.3];
                str = {"Angle = "+ sprintf('%.2f',angle(end)) + " deg",
"Torque = "+ sprintf('%.2f',torque(end)) + " N-m"};
                delete(a1);
                a1 =
annotation('textbox',dim,'string',str,'FitBoxToText','on', 'FontSize', 12,
'FontName', 'Times');
                drawnow;
                %fprintf(['Angle: %s' char(176) '\tTorque:
%sNm\n'],newStr(1),newStr(2));
            end
        end

        if emulate
            for c=1:length(dataE)
                angle = [angle;dataE(c,1)];
                torque = [torque;dataE(c,2)];
                xlim([0 max(angle)+5]);
                ylim([0 max(torque)+.2]);
                addpoints(livePlot,angle(end),torque(end));
                dim = [0.15 0.6 0.3 0.3];
                str = {"Angle = "+ sprintf('%.2f',angle(end)) + " deg",
"Torque = "+ sprintf('%.4f',torque(end)) + " N-m"};
                delete(a1);
                a1 =
annotation('textbox',dim,'string',str,'FitBoxToText','on', 'FontSize', 12,
'FontName', 'Times');
                drawnow;
                %fprintf(['Angle: %.2f' char(176) '\tTorque:
%.4fNm\n'],angle(end),torque(end));
                %pause(.05);
            end
            [rownum,colnum]=size(dataE);
            if colnum == 3
                loadCellE = dataE(3);
            end
            done = 1;
        end

    end
    if ~emulate
        clear arduino;
```

```
        fprintf("\nTest ended, re-run script for the next sample\n-=-=-=-=-=-=-
=-=-=-=-=-=-=-\n");
        open = 0;
    end
    delete(a1);
else
    fprintf("NO OPEN PORTS");
end
%% Fixing the offset
if ~emulate
    try
        TF=ischange(torque(1:40,1),'variance','Threshold',15); % Threshold
tuned on nice data
        breakID = find(TF,1);
        fprintf('The applied offset is %.2f deg\n',angle(breakID));
        torque = torque(breakID:end);
        angle = angle-angle(breakID);
        angle = angle(breakID:end);
    catch
        fprintf('Failed to offset data');
    end
end

%% Angle Vector Conversion from Degrees to Radians
angle_rad = angle.*(pi/180); % Angle conversion from deg to rad.

%% Sample Dimensions
d = 5e-3; % [in m]
r = d/2; % [in m]
l = 20e-3; % [in m]

%% Mechanical Properties of PLA (from Literature)
if(contains(material,"pla",'IgnoreCase',true))
    shearmodulus_lit = 700; % [in MPa]
    ultimateshear_lit = 39; % [in MPa]
    materialType = "PLA";
elseif(contains(material,"al",'IgnoreCase',true) ||
contains(material,"aluminum",'IgnoreCase',true))
    shearmodulus_lit = 26000; % [in MPa]
    ultimateshear_lit = 207; % [in MPa]
    materialType = "AL";
elseif(contains(material,"abs",'IgnoreCase',true))
    shearmodulus_lit = 680; % [in MPa] +-40
    ultimateshear_lit = 24.4; % [in MPa]
    materialType = "ABS";
else
    shearmodulus_lit = 700; % [in MPa]
    ultimateshear_lit = 39; % [in MPa]
    materialType = "UNKNOWN(using PLA props.";
end
%% Plotting of Torque vs. Angle of Twist
torque_smooth = smoothdata(torque,'gaussian'); % Smooth Torque Data

plot(angle,torque_smooth, 'k', 'LineWidth', 2)
title("Torque vs. Angle of Twist", 'FontSize', 18, 'FontName', 'Times')
xlabel("Angle of Twist (rad)", 'FontSize', 14, 'FontName', 'Times')
```

```
ylabel("Torque (N-m)", 'FontSize', 14, 'FontName', 'Times')

%% Plotting of Shear Stress vs. Shear Strain
shear_strain = (angle_rad .* r)/l;
shear_stress = (torque .* r)./((pi/32)*d^4);
shear_stress_MPa = shear_stress/(10^6);

figure
plot(shear_strain,shear_stress_MPa, 'k', 'LineWidth', 2, 'DisplayName',
'Sample Data')
title("Shear Stress - Shear Strain Curve", 'FontSize', 18, 'FontName',
'Times')
xlabel("Shear Strain (rad)", 'FontSize', 14, 'FontName', 'Times')
ylabel("Shear Stress (MPa)", 'FontSize', 14, 'FontName', 'Times')
hold on

% Identifying and Plotting Experimental Mechanical Properties:
    % Finding and Labelling Ultimate Shear Stress:
    ult_shear_stress = max(shear_stress_MPa);
    ult = find(shear_stress_MPa == ult_shear_stress);
    plot(shear_strain(ult),shear_stress_MPa(ult), '.r','MarkerSize',20,
'DisplayName', 'Ultimate Shear Stress')
    hold on
    text(shear_strain(ult) - 0.01, shear_stress_MPa(ult) - 1.5,
[num2str(shear_stress_MPa(ult)) ' MPa'], 'FontSize', 12.5, 'FontName',
'Times', 'Color', 'r')

    % Plotting the Offset Elastic Region, Finding and Labelling the Shear
    % Modulus
    [idx val] = max(shear_stress_MPa);
    Thresh = val;
    [TF S] = ischange(shear_stress_MPa, 'linear', 'Threshold', Thresh);
    idx = find(TF);
    offset = shear_strain(idx(1))*0.2;
    plot(shear_strain(1:idx(1))+offset, shear_stress_MPa(1:idx(1)), '--b',
'LineWidth', 1.5, 'DisplayName', 'Elastic Region')
    p = polyfit(shear_strain(1:idx(1))+offset, shear_stress_MPa(1:idx(1)),
1);
    shear_modulus = p(1);
    text(shear_strain(round(idx(1)/2))+offset+0.001,
shear_stress_MPa(round(idx(1)/2)), ['\rightarrow Shear Modulus = '
num2str(shear_modulus) ' MPa'], 'FontSize', 12, 'FontName',
'Times','Color','b')

    % Adding a Legend
    legend('Location', 'southeast', 'FontName', 'Times')

%% Percent Errors Associated with the Experimental Mechanical Property Values
err_shearmodulus = (abs(shearmodulus_lit -
shear_modulus)/shearmodulus_lit)*100;
err_shearstrength = (abs(ultimateshear_lit -
ult_shear_stress)/ultimateshear_lit)*100;

% Print the Results in a Message Box
if loadCellE
```

```
    err_LC = "Error Detected";
else
    err_LC = "No Error Detected";
end
EM = msgbox(sprintf('The percent errors associated with the experimental
shear modulus and ultimate shear stress are %f%% and %f%%,
respectively.\n\nLoad Cell Status: %s   Material:
%s',err_shearmodulus,err_shearstrength,err_LC,materialType));
EMT = findall(EM,'Type','Text');
EMT.FontSize = 16;
EMT.FontName = 'Times';
deltaWidth = sum(EMT.Extent([1,3]))-EM.Position(3) + EMT.Extent(1);
deltaHeight = sum(EMT.Extent([2,4]))-EM.Position(4) + 10;
EM.Position([3,4]) = EM.Position([3,4]) + [deltaWidth, deltaHeight];
EM.Resize = 'on';

%% Saving the file
if ~emulate
    FileName =
['TwistAngleTorqueSampleData_',material,'_',sprintf('%.2f',err_shearstrength)
,'E_',datestr(now, 'mmm-dd-yyyy'),'.mat'];
    dataSave = [angle(:,1) torque(:,1)];
    dataSave(1,3) = loadCellE;
    save(FileName,'dataSave') % Saves Data
    fprintf("\nData Saved as: %s\n", FileName);
end
```