An Arduino-based Uniaxial Material Testing Prototype

Christopher P. Kenaley
Department of Biology
Boston College
140 Commonwealth Avenue
Chestnut Hill, MA 02467, USA
kenaley@bc.edu
cpkenaley@gmail.com

Abstract

This document outlines how to assemble, run, and retrieve data from a prototyped uniaxial materials testing unit (MTU). The goal of this guide is to give researchers and scientists interested in material properties of biological tissues the wherewithal to reproduce the MTUs used in (Kenaley et al., Submitted) and modify the design as they see fit. These MTUs are intended for use in a broad array of settings, including use by senior scientists and graduate students in the laboratory as well as in engineering and biology classrooms in the context of activities that explore material properties.

Contents

1 Introduction

The design of the MTU prototype is anchored around the Arduino open-source hardware and software platform, in particular, the Uno R3 microcontroller programmed with Arduino's integrated development environment (IDE). The design goals of the MTUs were to prototype a uniaxial testing rig that was (1) platform independent, (2) low cost, and (3) capable of producing high-quality load and position data. Through Arduino's IDE, programming of the MTUs is possible on both Windows and Macintosh machines. By using relatively inexpensive Arduino Uno R3 processors (or their clones) that can be interfaced with low-cost expansion boards (i.e., shields) that read data from inexpensive Wheatstone-bridge load cells and control inexpensive linear-actuators, each MTU has a low-end cost of <$500 and high-end cost of <$1k.

2 Assembly

2.1 Parts list:

Item	model/make	max. cost
microcontroller	Arduino Uno R3	$20
linear actuator	Actuonix L12-P	$80
motor-control shield	MegaMoto Plus	$60
Load-cell amplifier shield	Robot Shop	$20
Load-cell	Futek LSB200, 2-lb JR	$500
3-D printed clamps	custom	$5
aluminum framing	80/20	$20
motor bracket	Firgelli	$10
metric breadboard	Edmunds optics	$150
jumper wires	Haitronic	$8
push buttons	Uxcell	$7
12V power supply	Mean Well	$50
	total:	$920

2.2 Parts

Microcontroler:

The MTU prototype is based upon the Arduino Uno R3 microcontroller. Many reliable clones are available. We've found the Elegoo to be reliable at nearly half the cost.

Model: Arduino Uno R3
Source (cost): Amazon (∼ $20)

Model: Elegoo UNO R3
Source (cost): Amazon (∼ $12)

Linear motors:

There are many linear motors to choose from, however, we found those with potentiometer position feedback from Firgelli and their spin-off, Actuonix, to come in a variety of capacities. Firgelli's large actuators have attractive speed (∼0.5–2 in/s), load (35–150 lbs), and precision (<0.15 mm over 4") specifications for large or very stiff materials. For smaller or less stiff materials, Actuonix provides a number of feedback-capable

linear actuators of lower capacities but much higher precision. We used the L12-P that produces up to 22 N of force, a max speed of 25 mm/s, and precision of 0.03 mm over a stoke of 30 mm .

Precision ($\mathcal{P}$) of the motors in mm was determined by the following:

$$\mathcal{P} = \frac{Pt_{max} - Pt_{min}}{S_{max} - S_{min}} \quad , \tag{1}$$

where Pt_{max} and Pt_{min} are the maximum and minimum potentiometer analog feedback values, respectively, and S_{max} and S_{min} are the maximum and minimum stroke extension in mm of the motor.

Brackets for the large motors can be bought from Firgelli; these permit mounting to a frame (see below). The small Actuonix motors are shipped with mounting hardware.

Model: Firgelli FA-PO-150-12 12-V DC, 4", 150-lb linear actuator
Source (cost): Firgelli ($\sim$ \$140)

Model: Actuonix L12-P 12-V DC, 30 mm, 22-N linear actuator
Source (cost): Actuonix ($\sim$ \$80)

Model: Firgelli MB6 mounting bracket for large linear actuators
Source (cost): Actuonix ($\sim$ \$80)

Figure 1: Linear actuators used in MTU prototypes: Firgelli FA-PO-150-12 12-V DC, 4", 150-lb linear actuator (left) and Actuonix L12-P 12-V DC, 30 mm, 22-N linear actuator (right)

Motor-control board:

One of the very attractive qualities of Arduino Uno platform is the myriad number of interfacing shields. Shields require no soldering and simple stacking into the digital and analog pins of the Arduino board. The Robot Power MegaMoto Plus acts as a full H-bridge to drives the motor with full variable-speed control both forward and reverse through pulse width modulation (PWM) from the Arduino. The shield requires a 12-V DC power source to drive the linear actuators.

Model: MegaMoto Plus motor control shield for Arduino
Source (cost): Jameco ($\sim$ \$60)

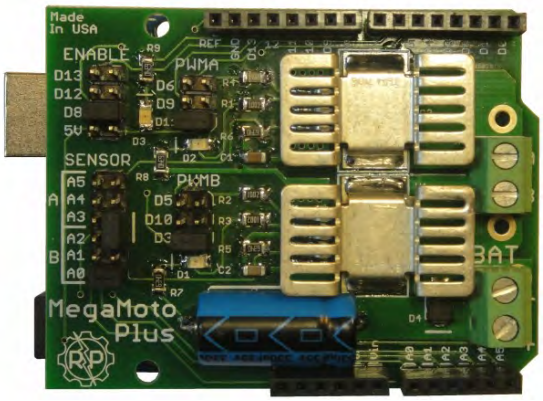

Figure 2: MegaMoto Plus motor control shield for Arduino.

Load-cell amplifier:

The voltage change that comes as the result of strain in the load cell (see below) is minute—often in the order of millivolts—and requires amplification. Fortunately, several shields or even breakout boards are available that amplify these small voltage changes. The Strain Gauge / Load Cell Amplifier Shield from Robot Shop was designed for precise amplification of measurements specifically from bridge amplifiers. This shield is preferred to a breakout board because no soldering or additional wiring is required. Based upon a AD8426 amplifier, this load-cell shield permits reference voltage adjustment and two instrument interfaces.

Model: Robot Shop Strain Gauge / Load Cell Amplifier Shield
Source (cost): Robot Shop ($\sim$ \$20)

Figure 3: Strain Gauge / Load Cell Amplifier Shield.

Load-cell: There are a few hardware options for measuring loads in the MTU, the basis of stress calculation, all based on Wheatstone bridge arrangements in "S"- or binocular-beam configurations **??**. When a voltage is passed through the bridge, the change in resistance of the strain gauges modifies the output voltage. Known loads can be used to calculate force or

mass based on this voltage. For uniaxial tests, either configuration, "S" type or binocular beams, are useful, and in preliminary comparisons with fish skin, we found no significant differences in precision or repeatability. Binocular beam types are much less expensive (<20), come in a wide array of load ranges and are very precise (in the range of ± 0.05%); however, safe overloads are on the order of 100–250%. "S"-type load cells are much more expensive (>$200), at least as precise and accurate, but typically have higher safe overloads that approach 1000%. In addition, "S"-beam load cells are much more compact at similar load ranges. Both are defensible choices, however, if you are unsure of the material's stiffness (i.e., it could be really stiff) or space is an issue, "S" type load cells may be a better option. Loads for skin samples 1–2 cm wider required load cells with capacities around 1–2 Kg, wheres samples wider than this require stiffer load cells of 5 Kg or more.

Figure 4: A binocular-beam load cell (left) and "S"-beam load cell from Futek (right)

a

Model: Phidgets 5-Kg Micro Load Cell
Source (cost): Robot Shop (∼ $7)

Model: Futek LSB200, 2-lb JR S-Beam Load Cell
Source (cost): Futek (∼ $500)

Clamp:

The skin, or whatever other tissue, to be tested must be secured to the motor end and load cell. There are several options, including milling clamps from ABS plastic or aluminum and 3D printing. Both options are inexpensive, assuming a 3D printer is readily available, however, the latter is much less labor intensive. The simple clamp design in Figure 6 can be downloaded and modified. For each prototype, we printed a set of clamps with PLA. A 2.5-mm hole in the base was threaded with an M3 so that a bolt passed through the cap and tightened in the base cinched the material. To bind a clamp to the force transducer and motor end, the rear face of each clamp was as also tapped to accept a metric post that could be screwed into M2–4 threaded holes of the transducers and a tapped hole at the motor end Figure 5.

Model: Custom 3D-printed clamps
Source (cost): STL on GitHub directory (∼ $1–5)

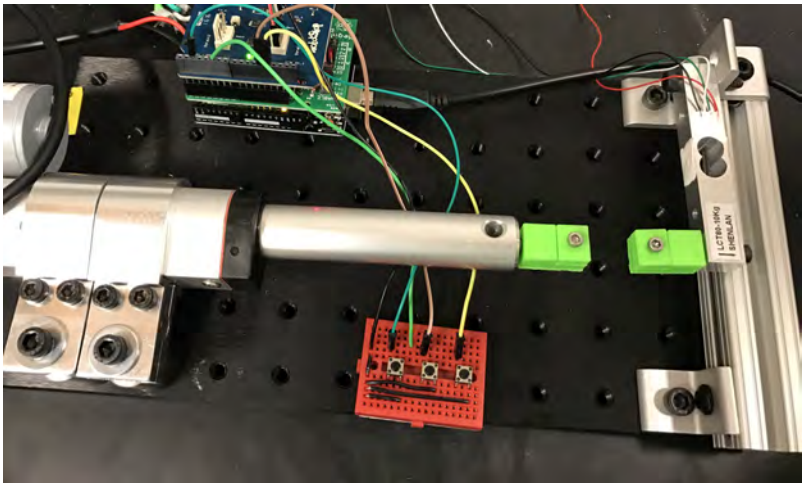

Figure 5: An assembled prototype.

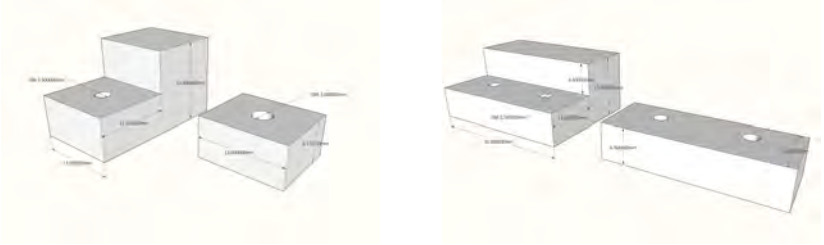

Figure 6: CAD models of small (left) and large (right) MTU clamps.

Miscellaneous parts: The motor and load cell must be aligned and mounted to a rigid structure. For this, aluminum framing works well. We used 10 series 1"x1" slotted framing from 80/20 bolted to an Edmund Optics 30-cm x 15-cm breadboard. 80/20 will ship custom lengths for their framing, thus avoiding any need to cut the aluminum yourself.

Wiring of the prototype can be done with solid-core prototyping wire or with jumper wires that come in a variety of lengths. Using jumpers avoids the need for wire stripping. In addition, to make wiring easier across what can be crowded surface, we used a solderless breadboard. This also permitted the installation of a hardware interface with which the user can adjust the motor length and initiate the program with three tactile push buttons (Figure 5).

Lastly, a 12-V DC power source must supply the motor. Very inexpensive models with outputs < 1 A are option if only 1 small motor is to be powered. We chose to power several MTUs at once with large motors that draw up to 5 A and hence required a power supply (30 A) that could handle the increased load.

Model: 80/20 10 series 1"x1" aluminum framing and brackets
Source (cost): 80/20 ($\sim$ \$20)

Model: Edmund Optics 300 mm x 150 mm breadboard
Source (cost): Edmund Optics ($\sim$ \$150)

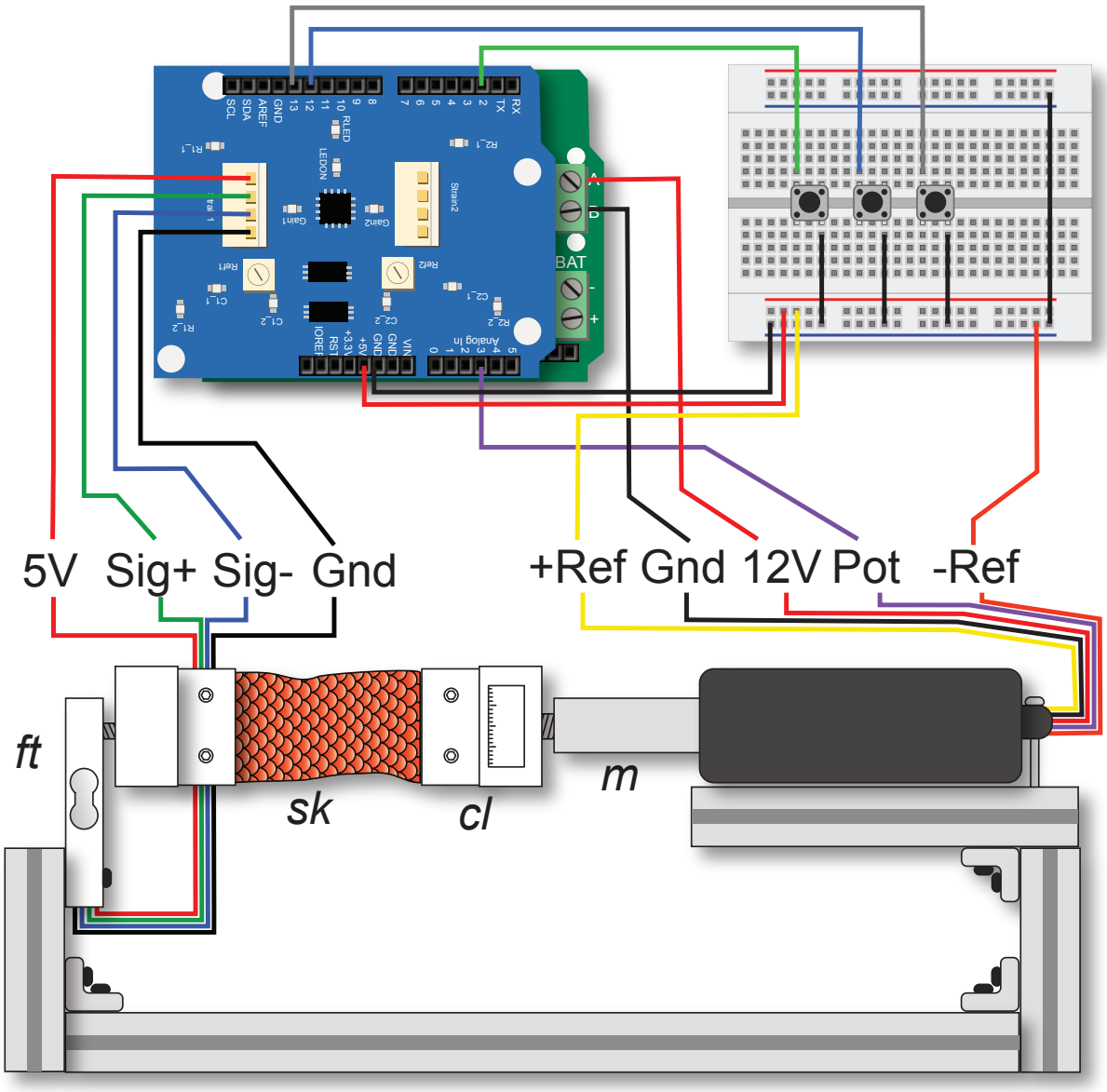

Figure 7: Schematic diagram of an MTU prototype.

Model: Haitronic 120-peice, 20-cm Jumper Wires
Source (cost): Amazon (∼ $8)

Model: Mini solderless Breadboard
Source (cost): Amazon (∼ $8)

Model: Uxcell 12x12-mm Momentary Tactile Push Button Switch
Source (cost): Amazon (∼ $7)

Model: Mean Well NES-350-12 12V 350 Watt Power Supply
Source (cost): Amazon (∼ $50)

2.3 Assembly process

Framing and mounting:

We found that securing the large Firgelli motors was easiest using their MB6 bracket bolted directly to a breadboard. The smaller Actuonix motors were mounted to 1"x1" 80/20 pieces using inside-corner brackets (Figure 8). The load cells were mounted to inside-corner brackets by thread posts screwed into the load cell and tapped holes in the bracket (Figure 7). The bracket-load cell unit was then secured to a piece of 80/20 and aligned with the motor. A clamp was then secured to the motor end and load cell via threaded posts.

Stacking shields:

The motor control shield is stacked onto the Arduino first, then the load-cell amplifier shield is stacked on the motor-control shield. This must be done in this order to insure that there is access to the load-cell interface pins on the top of the amplifier shield. Access to the motor-control shield must be maintained for wiring, thus the amplifier shield should be stacked after wiring is complete.

Wiring:

Wiring the prototype as presented in Figure 7 was relatively simple. The following instructions assume all hardware is in place and mounted, including the framing, motor, and load cell. When wiring the prototype, it must be kept in mind that several pins on the Arduino are dedicated to the motor-control (digital 3,8 , and 11) and amplifier shields (analog 0 and 1).

1. Wire power supply to motor-control shield. It's important that the polarity is not switched:
   (a) Confirm it is unpowered (i.e., off!).

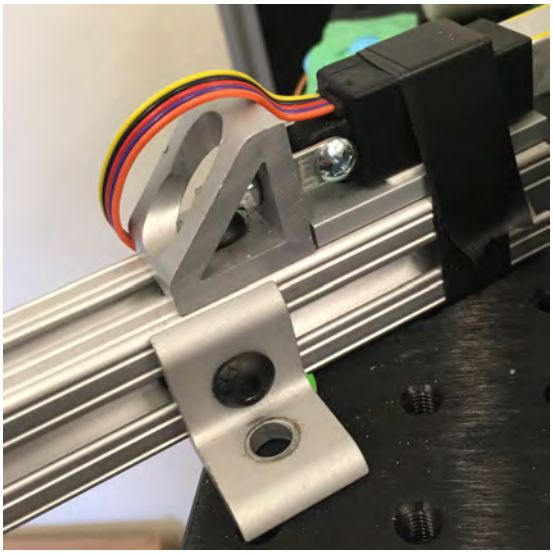

Figure 8: Mounting a small DC linear actuator from Actuonix to 80/20 framing and a metric breadboard.

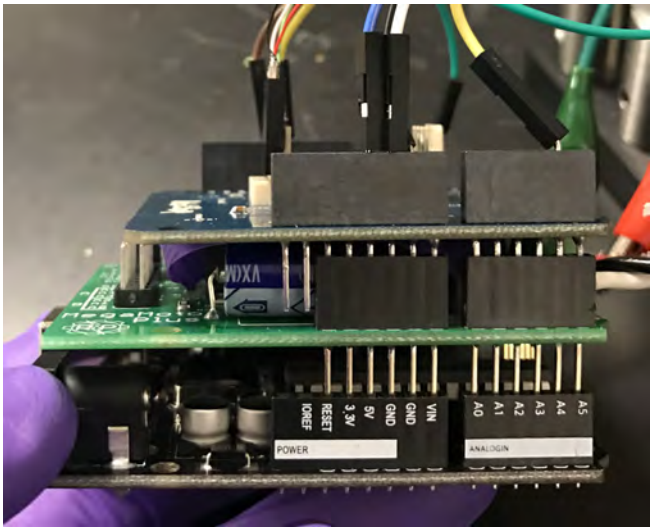

Figure 9: The Arduino Uno R3 with stacked motor-control and load-cell amplifier shields.

   (b) +12V to BAT+

   (c) Ground to Bat-

2. Install three push buttons on the breadboard spanning the board's L/R sides.

3. Using a jumper or short piece of solid-core wire, connect the negative rails of the bread board and then the same leg on the same side of each button to the negative rail.

4. Using a jumper, connect the negative rail to the GND pin and the positive rail to the +5V pin on the Arduino-shield stack.

5. Wire the actuator to motor-control shield:

   (a) 12V (red) to A terminal

   (b) Gnd (black) to B terminal

6. Wire the actuator to breadboard and Arduino-shield stack:

   (a) +Ref, potentiometer feedback positive reference (yellow), to positive rail of the breadboard

   (b) -Ref, potentiometer feedback negative reference (white or orange), to negative rail of the breadboard

   (c) Pot, potentiometer feedback wiper signal (blue or purple) to A3 of the top shield.

7. Wire the load cell to Arduino-shield stack:

   (a) +5V, positive excitation/VCC (red), to pin 1 (closest to digital pins of shield) on strain1 connection

   (b) +Sig, positive signal /output(green or blue), to pin 2 on strain1 connection

   (c) -Sig, negative signal/ouput (white), to pin 3 on strain1 connection

(d) Gnd, negative excitation/ground (black), to pin 4 on strain1 connection

8. Wire buttons to Arduino-shield stack. Wires or jumpers must be connected to the leg on the button that extends from the opposite side and corner Figure 7:
   (a) connect digital pin 2 to the first button (this will be the "go" button that initiates movement and data collection)
   (b) connect digital pin 12 to the second button (this will be the "pull" button that positions the motor before testing)
   (c) connect digital pin 13 to the third button (this will be the "push" button that positions the motor before testing)
9. Connect USB cable to computer and microcontroller

3 Programming

Once assembled, a program (i.e., sketch) must be uploaded to the microcontroller. Programming in the Arduino IDE is relatively straight forward and well supported by Arduino and advice and tutorials offered by hacker and DIY sites online (e.g., Instructables, Sparkfun, etc.). The sketch provided with this document will allow the user to position a motor before testing—perhaps to appropriately tension or condition the material—and initiate at least 1 cycle of tensile extension and compression with continuous load sensing.

The sample rate is determined by the data transfer rate of the serial connection (the baud rate) and the amount of data sent by the microcontroller. At a baud rate of 57600, the example sketch samples at ∼100 Hz, i.e., every 10 ms.

Setting program parameters:

Several variables early in the sketch control important testing parameters and calculations within the program:

1. `tol`: Tolerance in pot units for the goal positions. Set too low, the program will run poorly and possibly miss a goal position. Set too high, and the precision in meeting the goal positions is compromised.
2. `per`: Strain in decimal form (i.e., strain %/100) or how far you want the motor to shorten and lengthen the tissue specimen.
3. `relax`: The time in ms for stress relaxation between extension and compression. A low value below ∼50 ms will result in a very smooth transition between motor retraction and extension.
4. `motSpeed`: Controls the velocity of the motor. This value corresponds to the duty cycle parameter passed to the microcontroller with `analogWrite`. This value ranges from 0 to 255, with low values corresponding to slower speeds, higher to higher speeds. The user should perform a set of experiments to calculate the relationship between this value and speed. Motor speed with determine strain rate, a parameter that can alter the stiffness of viscoelastic materials. One could implement a motor-speed variable that is determined by the length of the specimen so that the speed translates to similar strain rates between samples of different lengths.

```
float tol = 4;//tolerance of stop positions (+/- tol) in pot
     units
const float per = .15; // strain percentage/100
int count = 5;//number of cycles
int relax = 50; //time (ms) to pause after pull, set low
    (<50 ms to have quick switch from pull to push)

//speed of motors. Will have to alter according to sample
    length
//to achieve your desired strain rate. May very with motor
    model
int motSpeed = 150;
```

Motor and load-cell calibration:

The next two blocks of code contains values for calibration of motor position and load cell values. This calibration process should be performed after assembly. Motor calibration must be performed each time the mounted position of the motor changes relative to the load cell (e.g., it is moved along the framing to accommodated a larger or smaller specimen). Two-point load-cell calibration should be performed regularly to ensure good accuracy. Hanging known masses from the load cell is quick and easy.

1. `len`: Distance between the clamps when the motor is fully extended, that is, the smallest sample length. We used digital calipers to measure the distance between the faces of the clamp. As the motor changes position, the sketch with add the motors displacement to this value.
2. `potMax`: Maximum value of motor potentiometer reading. This value is printed to the serial monitor under "pot val." Run the program, extend the motor fully, and enter the value attained at full extension.
3. `potMin`: Minimum value of motor potentiometer reading. Retract the motor fully, and enter the value attained.
4. `potMax` and `potMin`: Maximum and minimum positions, respectively, of the motor relative to a reference (e.g., the motor collar).
5. `ReadingA_Strain1` and `ReadingB_Strain1`: Analog reading from the load cell under the first and second load, respectively. Raw values for the load cell are printed under "strain" in the serial monitor.
6. `ReadingA_Strain2` and `ReadingB_Strain2`: The values of the first and second know masses, respectively, suspended from the load cell in grams.

```
////************ calibration values for motor position (
    calipers help!!) ****************//////
float len = 6.9;// distance between clamps when motor is
    fully extended (mm)
float potMax = 862; //max pot reading
float potMin = 12; //min pot reading
```

```
float strokeMax = 115.6; //max position (mm) of motor from a
     reference point (e.g., collar of housing)
float strokeMin = 19.5; // min position (mm) of motor from
    same reference point

////************ calibration values for load cell
    ****************//////
float ReadingA_Strain1 = 22.0;//first raw analog reading
float Load_Strain1 = 50; // first load (Kg,lbs..)
float ReadingB_Strain1 = 489;//second raw analog reading
float LoadB_Strain1 = 200; // second (Kg,lbs..)
```

References

Kenaley, C. P., A. Sanin, J. Ackerman, and J. Yoo, Submitted. Skin Stiffness in Ray-finned Fishes: Contrasting Material Properties Between Species and Body Regions. Journal of Morphology .