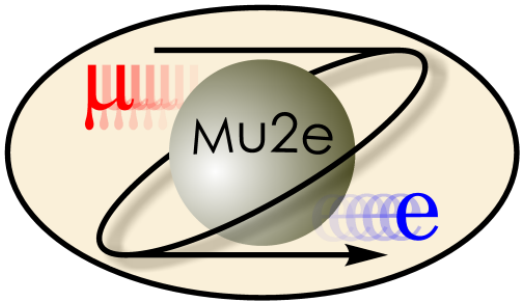

Automated Sense Wire Tensioner: Project Status Update 2

Cole Kampa

University of Minnesota

March 8, 2018

kampa041@umn.edu

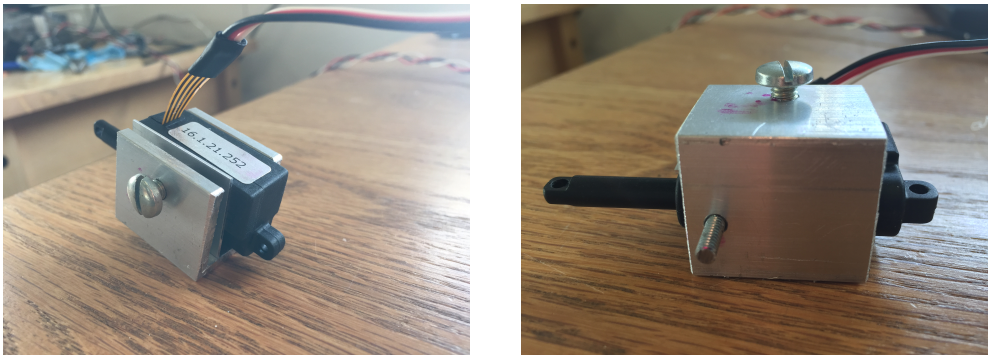

Figure 1: A U-channel piece of aluminum houses the actuator. A set screw tightens a thin sheet metal plate against the actuator. A screw is threaded through the bottom in place of a dowel pin (right).

Recent Updates

- **Motor and Load Cell Mounts:** Finished prototype mounts for actuator (figure 1) and load cell (figure 2).
- **First Panel Test:** Tested tensioning a thread with the test panel using the existing pulley plate holes (setup shown in figure 3).
- **Planned Motor Replacement:** Switching the current motor to a more precise stepper motor should allow us to remove the weak spring and maintain a desired precision, which will simplify positioning (spring may sag). New motors and drivers have been ordered (1 week lead time).
- **Thread Gripping Method:** An alligator clip with the protective tubing still on the ends can grip the thread enough to hold up to 400gf (figure 4). This is a quicker attachment than the previous loop and hot glue method. When the stepper motor arrives, we will have to find a way to fix the alligator clip to the linear motion piece of the motor.

To-Do

- Aruino Code:

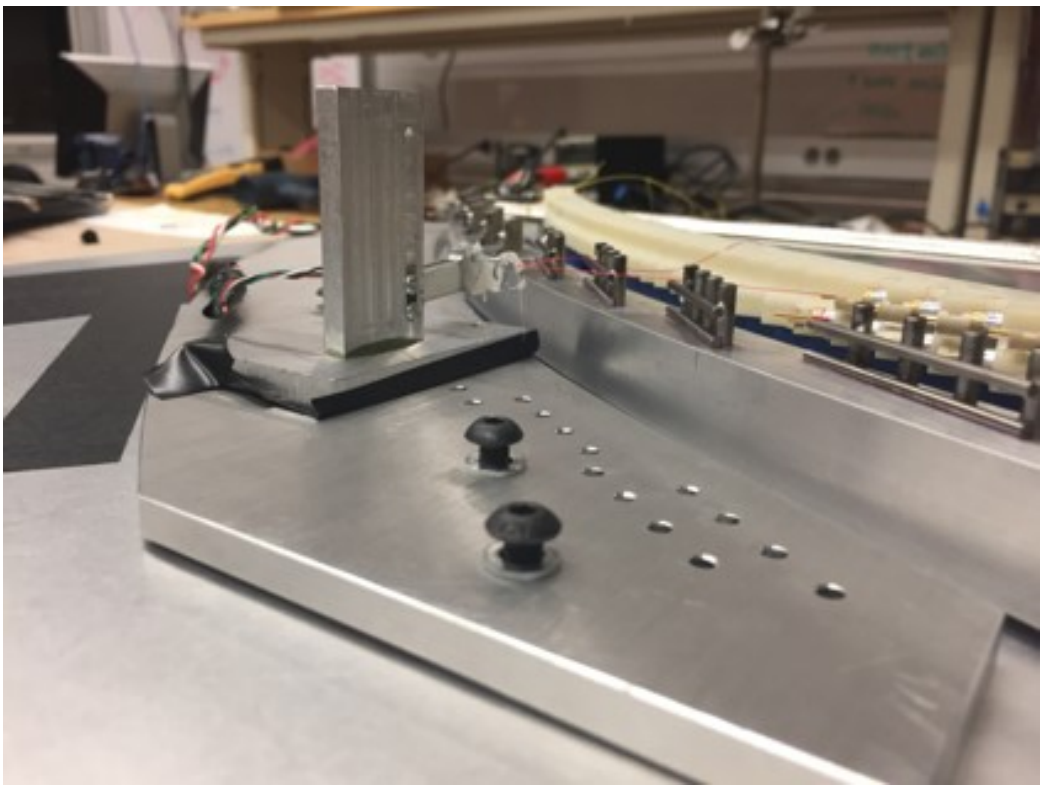

Figure 2: The load cell is fastened on one end to an aluminum upright epoxied with DP-810 to the aluminum base. A screw is used in place of a dowel pin. The screw fits into the positioning holes on the pulley plate.

- Integrate the new stepper motor (20DAM10D1B-K) + stepper driver (ROB-12779) [3 of each ordered on 3/7/18, send one set to Aseet at FNAL]
  - Find proper Arduino library for steppers and work to find the best way to control the stepper (feed in number of steps and speed)
  - A startup method must be in place to send the stepper back to a desired initial position. This will happen anytime the Arduino is reset.
- Add calibration to main script for scheduled calibrations without changing the code
  - Can follow example in `SparkFun_HX711_Calibration` script

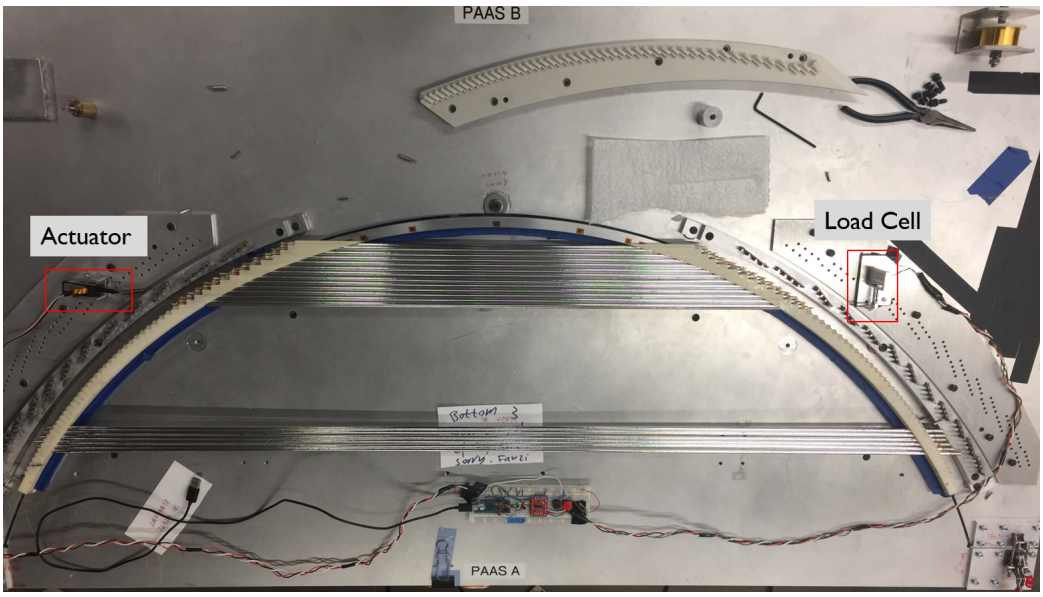

Figure 3: Setup of the first panel test for ergonomics of tensioning.

i.e. calculate tension with current calibration factor, iterate with a conditional to adjust calibration factor until proper calculated tension is met.

  - Adjust peripherals (LEDs, Buzzer)
    - Right now, light indicates good or bad tension but LEDs are dim. Three buzzes if 80gf is never met. Add a long buzz for "ready to solder ".
  - Do not allow motor to oscillate about 80gf.
    - Because of the hysteresis in the sense wire, we want to pull to tension but never release the tension

- Hardware:
  - Design mounts for load cell & stepper motor with two dowel pins (two will help set the rotational position of the load cell and motor for accurate tension measurements)
    - Should make drawings to have these machined

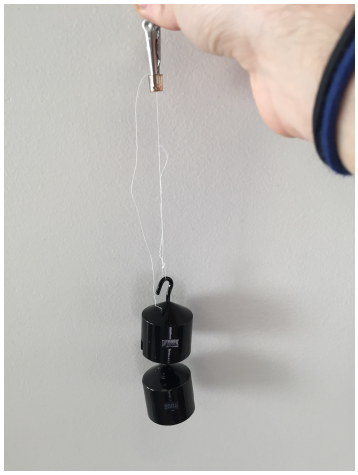

Figure 4: An alligator clip with protective tubing on the jaws can hold the thread up to 400 gf (left). Further designs must be made to attach the clip to the motor (right).

    - The pulley plates will need to be adjusted to suit the second hole at each position. There is room to move the holes farther from comb if more space is needed.
- Thread and wire holding methods
    - The thread can be held by alligator clips with protective plastic sleeves. Now we just have to find a way to mount the clip to the motor (see figure 4).
    - Wire should be fixed to load cell end. No testing has been done with the alligator clips. The current method involves clamping the wire with a strong spring fixed to PAAS B.

1 Arduino Code

The following code (`sense_wire_tension_v2_0.ino`) is currently being used.

```
// sense wire tension v2.0
// cole kampa -- kampa041@umn.edu
// arduino micro

#include <Servo.h>
#include "HX711.h"
```

```
7
8  //---actuator data--- (send a pulse of certain width to send
       actuator to a given position (in degrees, using Servo library
       )
9  #define actuatorPin 9
10 //---load cell--- (this is to the load cell amplifier, match
       load cell colored wires to color labeled pins on load cell
       amp)
11 #define loadDAT 3
12 #define loadCLK 2
13 //---LEDs--- (replace resistors or get different LEDs to make
       them brighter)
14 #define lowLED 5
15 #define goodLED 6
16 #define highLED 7
17 //---buzzer---
18 #define buzzerPin 12
19 #define buzz_delay 200  //defines period of buzzer
20 #define freq 220        //sets tone of the buzzer when a
       measurement is bad
21
22 //---tension definitions---
23 #define tension_good 80.0
24 #define tension_low 79.5
25 #define tension_high 80.5
26
27 //---linear equation definitions--- following y = m*x + b
28 //****tension_plot.pdf****
29 //two solutions for the low tension linear regime and high
       tension linear regime (cut at 31.0 gf)
30 #define low_tension_m -0.2214
31 #define low_tension_b 129.36
32 #define high_tension_threshold 31.0 //specified minimum force by
        spring manufacturer, verified by consistency tests
33 #define high_tension_m -1.5997
34 #define high_tension_b 172.83
35
36 //---reversing button-------------------------//
37 int startButton = 8;         // the number of the input pin
38 int state = HIGH;      // the current state of the output pin
39 int reading;           // the current reading from the input pin
40 int previous = LOW;    // the previous reading from the input pin
41 // the follow variables are long's because the time, measured in
        miliseconds, will quickly become a bigger number than can be
        stored in an int.
```

```
42 long time = 0;         // the last time the output pin was
       toggled
43 long debounce = 200;   // the debounce time, increase if the
       output flickers
44
45 //---initialize objects for actuator and load cell
46 Servo actuator;
47 HX711 load_cell;
48
49 //---other variable definitions
50 float tension_before = 0.0;
51 float tension_after = 0.0;
52 int position_last = 45;
53 char inByte;
54 String mode = "RESET";
55 //-----------------------------------------//
56 void setup() {
57   Serial.begin(9600);
58   Serial.println("wire tensioner v1.0:");
59
60   load_cell.begin(loadDAT, loadCLK);
61   load_cell.set_scale(-11690.f);
62   load_cell.tare();
63
64   pinMode(lowLED, OUTPUT);
65   pinMode(goodLED, OUTPUT);
66   pinMode(highLED, OUTPUT);
67   pinMode(buzzerPin, OUTPUT);
68   pinMode(startButton, INPUT);
69
70   actuator.attach(actuatorPin);
71   actuator.write(position_last);
72
73   digitalWrite(lowLED, HIGH);
74   digitalWrite(goodLED, HIGH);
75   digitalWrite(highLED, HIGH);
76   delay(2000);
77   digitalWrite(lowLED, LOW);
78   digitalWrite(goodLED, LOW);
79   digitalWrite(highLED, LOW);
80 }
81 //-----------------------------------------//
82 void loop() {
83   tension_before = load_cell.get_units(); //could add some
       number between get_units(##) to take an average of ##
```

```
   measurements
84 //allow serial input for debugging
85 if (Serial.available()) {
86   inByte = Serial.read();
87   if (inByte == '0') {
88     mode = "RESET";
89   }
90   if (inByte == '1') {
91     mode = "TENSION";
92   }
93 }
94 //---mode change button---
95 reading = digitalRead(startButton);
96 // if the input just went from LOW and HIGH and we've waited
    long enough to ignore any noise on the circuit, toggle the
    output pin and remember the time
97 if (reading == HIGH && previous == LOW && millis() - time >
    debounce) {
98   if (mode == "RESET")
99     mode = "TENSION";
100  else
101    mode = "RESET";
102  time = millis();
103 }
104 //the state measured in this loop becomes 'previous' for the
    next loop
105 previous = reading;
106
107 if (mode == "RESET") {
108   if (position_last != 45) {
109     actuator.write(45);
110     digitalWrite(lowLED, LOW);
111     digitalWrite(goodLED, LOW);
112     digitalWrite(highLED, LOW);
113   }
114   position_last = 45;
115   delay(500); //can increase delay to save processing
116 }
117 if (mode == "TENSION") {
118   position_last = actuator_adjust(position_last,tension_before
    );
119   if (position_last >= 140) {
120     mode = "RESET";
121     //buzz 3 times based on frequency and duration defined
    above
```

```
122         tone(buzzerPin,freq);
123         delay(buzz_delay);
124         noTone(buzzerPin);
125         delay(buzz_delay);
126         tone(buzzerPin,freq);
127         delay(buzz_delay);
128         noTone(buzzerPin);
129         delay(buzz_delay);
130         tone(buzzerPin,freq);
131         delay(buzz_delay);
132         noTone(buzzerPin);
133       }
134     }
135
136     tension_after = load_cell.get_units();
137
138     Serial.print("Tension: 1) ");
139     Serial.print(tension_before);
140     Serial.print("g, 2) ");
141     Serial.print(tension_after);
142     Serial.print("g; Mode: ");
143     Serial.print(mode);
144     Serial.print("; Position: ");
145     Serial.println(position_last);
146
147     delay(250);
148 }
149
150 //-------------------------------------//
151
152 int actuator_adjust(int position_old, float tension){
153     int position_new;
154     int delay_time = 800; //default delay for incrementing by 1
155     float delta_tension;
156
157     //new method for first time through based on linear equation
        definitions
158     if (position_old == 45) {
159       if (tension < high_tension_threshold) {
160         position_new = floor(low_tension_m * tension +
      low_tension_b);
161       }
162       else {
163         position_new = floor(high_tension_m * tension +
      high_tension_b);
```

```
164     }
165     delay_time = 4000; //5 s delay to start, adjust as necessary
166   }
167   //all other times, only increment +-1 and measure the tension
        again
168   else {
169     if(tension < tension_low) {
170       position_new = position_old + 1;
171       digitalWrite(lowLED, HIGH);
172       digitalWrite(goodLED, LOW);
173       digitalWrite(highLED, LOW);
174     }
175     else if (tension > tension_high) {
176       position_new = position_old - 1;
177       digitalWrite(lowLED, LOW);
178       digitalWrite(goodLED, LOW);
179       digitalWrite(highLED, HIGH);
180     }
181     else {
182       position_new = position_old;
183       digitalWrite(lowLED, LOW);
184       digitalWrite(goodLED, HIGH);
185       digitalWrite(highLED, LOW);
186       delay_time = 50;
187     }
188   }
189   //adjust position if the motor trys to overrun the bounds (45
        is full retraction, 140 is full extension)
190   if(position_new < 45) {
191     position_new = 45;
192   }
193   else if (position_new > 140) {
194     position_new = 140;
195   }
196   actuator.write(position_new);
197   delay(delay_time);
198   return position_new; //position_new becomes postion_last
199 }
```