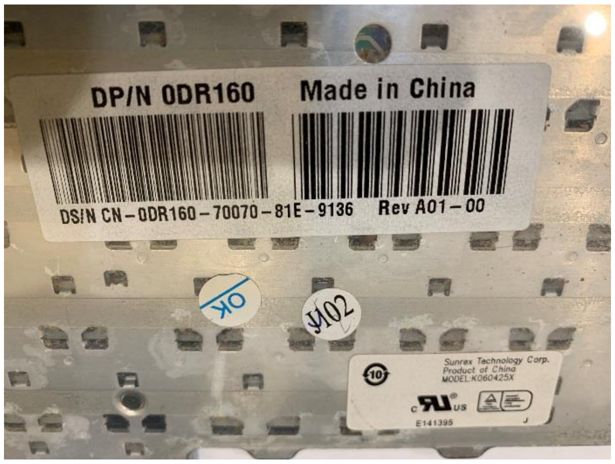
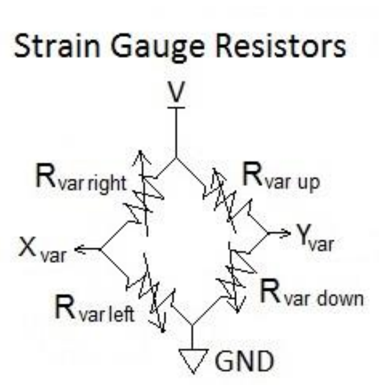
Dell D630 Pointing Stick Interfaced with a Teensy 3.2

The Dell D630 has a pointing stick (aka trackpoint) in the center of the keyboard as shown below.

The keyboard part number is given below.



The blue eraser head is attached to four resistive strain gauges configured as follows.



Unlike a T61 Thinkpad trackpoint which includes a PS/2 circuit, the Dell pointing stick is a “raw” strain gauge with 4 connection points; power, ground, X, and Y. The Dell motherboard receives these signals on an FPC cable and routes them to the touchpad electronics for conversion to PS/2.

Fluke 87 ohm meter measurements on the raw pointing stick signals are given in the table below. Each resistor is about 4KΩ.

V to GND	3.982KΩ
X to Y	3.977KΩ
X to GND	2.981KΩ
X to V	2.911KΩ
Y to GND	3.029KΩ
Y to V	3.061KΩ

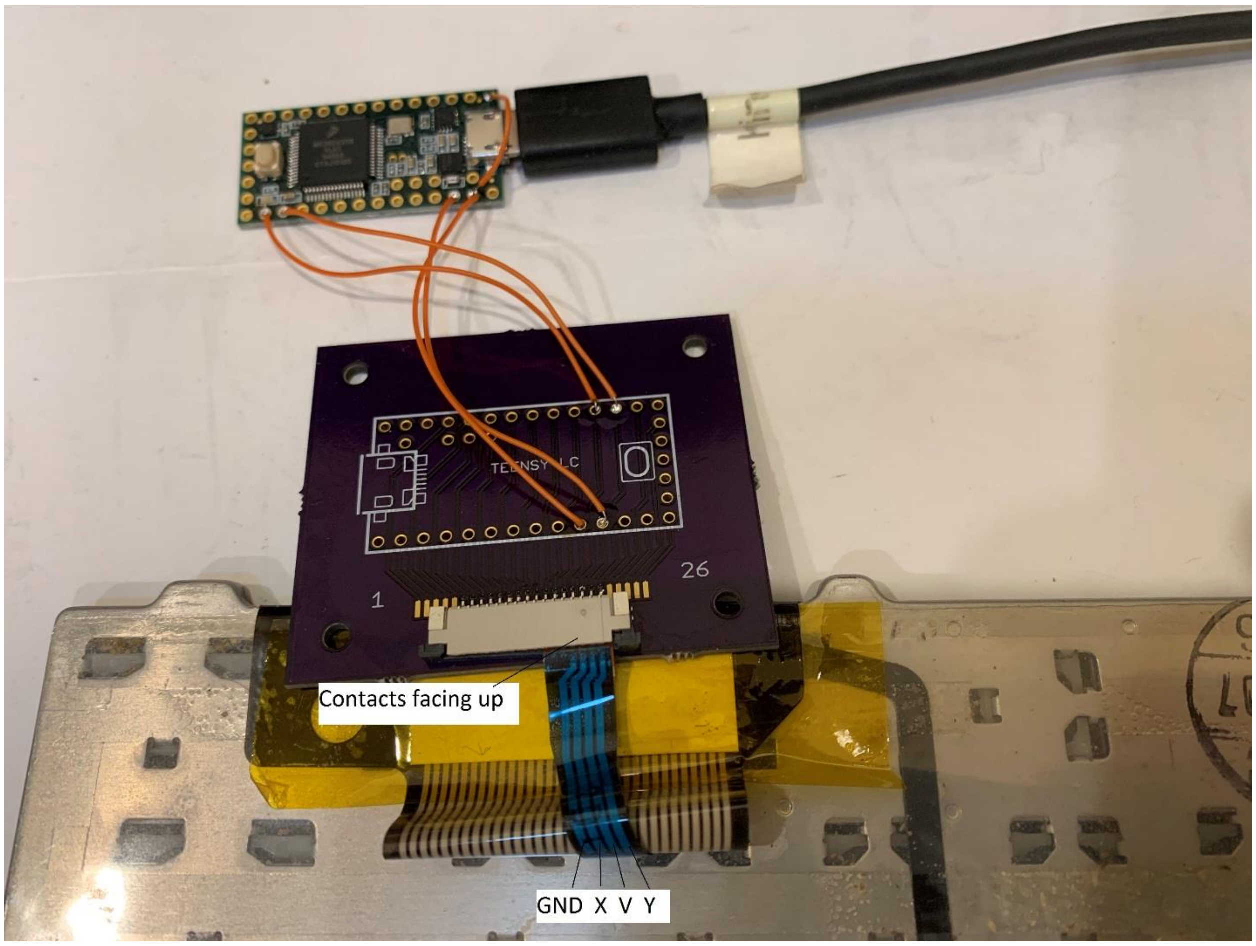
Applying 3.3 volts across V and GND gives the following voltages on the Fluke 87.

Vx	1.684V	Resting (no pressure)
Vy	1.641V	Resting (no pressure)
Vx	1.715V	Max pressure right
Vx	1.655V	Max pressure left
Vy	1.670V	Max pressure up
Vy	1.595V	Max pressure down

Max pressure changes the voltage plus or minus 30mv from the resting position (roughly).

I connected the X and Y signals to the A0 and A1 inputs of a Teensy 3.2. These inputs go to the internal 16 bit A to D converter. The voltage measurements shown above did not change, indicating the input impedance of the Teensy ADC is high.

I used an old circuit board with an FPC connector for attaching the cable (see below).



This also shows how the keyboard FPC cable and the pointing stick FPC cable have been taken out of the solderless connector used on the motherboard. I had to pull off the extra thick plastic backing on both FPC cables. The pointing stick cable had its sides trimmed with scissors and then two pieces of "stick-up" paper were attached to bring the thickness back to the normal 0.3mm so a generic connector could be used.

The Teensy 3.2 code "Dell_D630_Pointing_Stick.ino" sets the ADC to 16 bit resolution and averages 16 conversions to help with noise. The resting voltage of the X and Y strain gauges is stored at startup. In the main loop, if either the X or Y voltage is beyond the noise zone threshold, the Mouse.move function sends movement commands over USB. Experimentation showed that the noise zone value needed to be greater than 130. I used 175 for added margin. Experimentation also showed that the X and Y values sent over USB needed to be divided by a factor of 10 to slow down the cursor response.

If integrating this code with a keyboard scanning routine, provisions should be made to re-capture the resting X and Y voltage. This might be needed if the cursor starts to move without any pressure applied to the pointing stick (probably because of temperature changes). Watching for a function key sequence such as Fn-F8 could trigger new resting values to be captured.