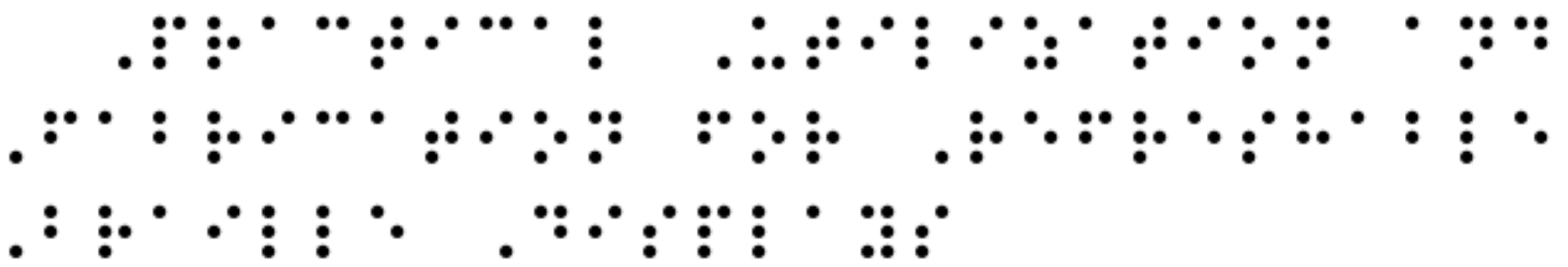
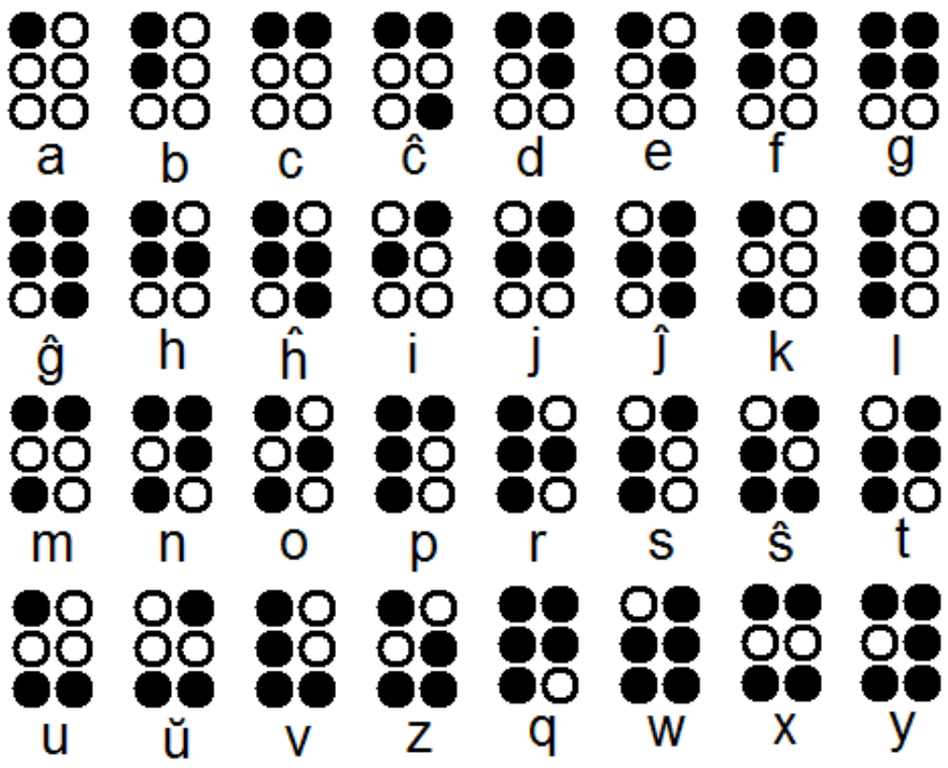
Practical Utilization and Fabrication for Refreshable Braille Displays



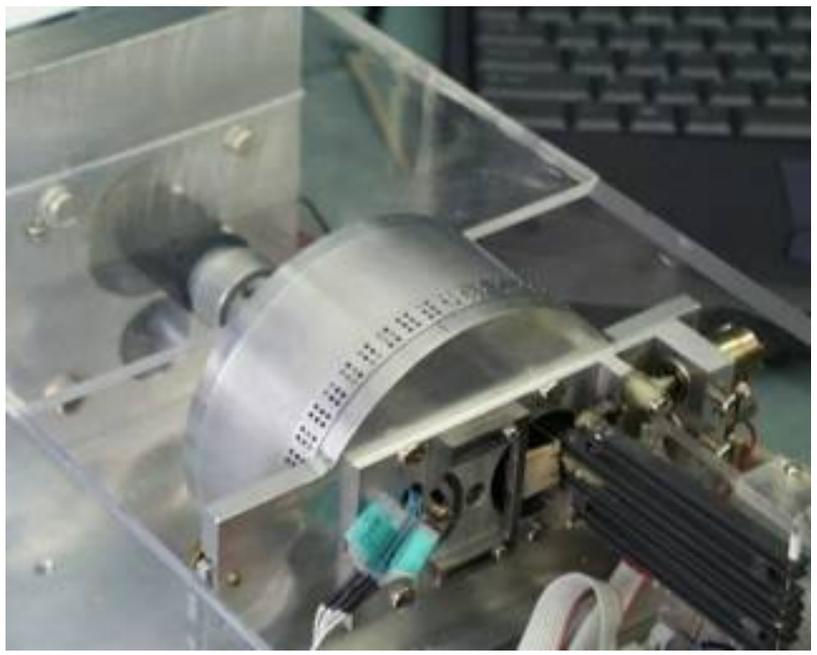
Definition

Braille was invented by Louis Braille as a type of night writing. It consists of cells with six raised dots. A refreshable Braille display is a device used by blind computer users for displaying a computer screen by means of round reading pins pushing through holes.



History

A rotating-wheel Braille display was developed in 2000 by the National Institute of Standards and Technology and another at the Leuven University in Belgium. Both wheels are not currently being commercialized. In these units, Braille dots are put on the edge of a spinning wheel, which allows the user to read continuously with a stationary finger while the wheel spins at a selected speed. The Braille dots are set in a simple scanning-style fashion as the dots on the wheel spin past a stationary actuator that sets the Braille characters. As a result, manufacturing complexity is greatly reduced and rotating-wheel Braille displays, when in actual production, should be less expensive than traditional Braille displays.



Problem

Visual disability is a widespread problem that affects 285 million people globally. Many of these people are blind and actively use Braille to read. Technology such as the iPhone has advanced while the technology used for people with visual disabilities or disabilities in general has gone almost nowhere. Also, technology for helping blind people “see” or feel pictures is wholly undeveloped.

Solution

The goal of this project is to create a display that can arrange pins in patterns to imitate the Braille alphabet and show images for blind readers.

How to move the pins?

- electromagnets
- piezoelectric
- memory alloys
- microfluidics
- lasers

The memory wire was chosen because electromagnets were too big, I couldn't figure out a way to use piezoelectricity, microfluidics were too big as well, and lasers would use too much heat.

Proof of concept

- instead of starting big with a screen full of characters, I'll start with making one character
- I can control whether there is electricity going through the circuit using the Arduino and transistors and therefore control the display

- Supplies
  - 6 pieces of 40 mm or 4 cm long 075 memory wire
  - 12 Dupont pins
  - 2 3D-printed boards
  - 6 3D-printed Braille pins
  - Breadboard
  - Arduino
  - 6 BC 547B E General Purpose Transistors
  - 6 216 Ω Resistors

Braille Dot Specifications

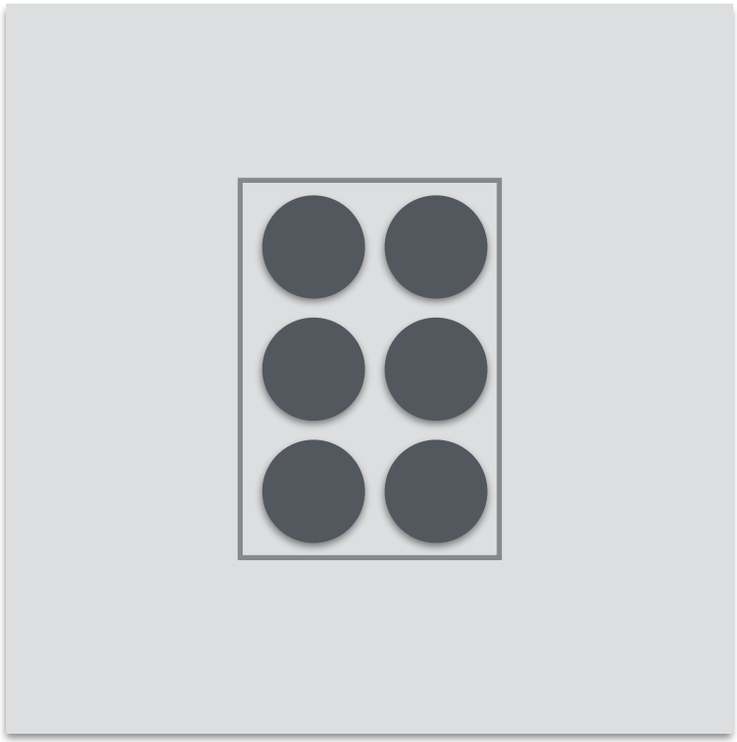
Measurement Range	Minimum in inches Maximum in inches
Dot Base Diameter	0.059 (1.5mm) to 0.063 (1.6mm)
Distance between two dots in the same cell	0.090 (2.3mm) to 0.100 (2.5mm)
Distance between corresponding dots in adjacent cells	0.241 (6.1mm) to 0.300 (7.6mm)
Dot height	0.025 (0.6mm) to 0.037 (0.9mm)
Distance between corresponding dots from one cell directly below	0.395 (10.0mm) to 0.400 (10.2mm)

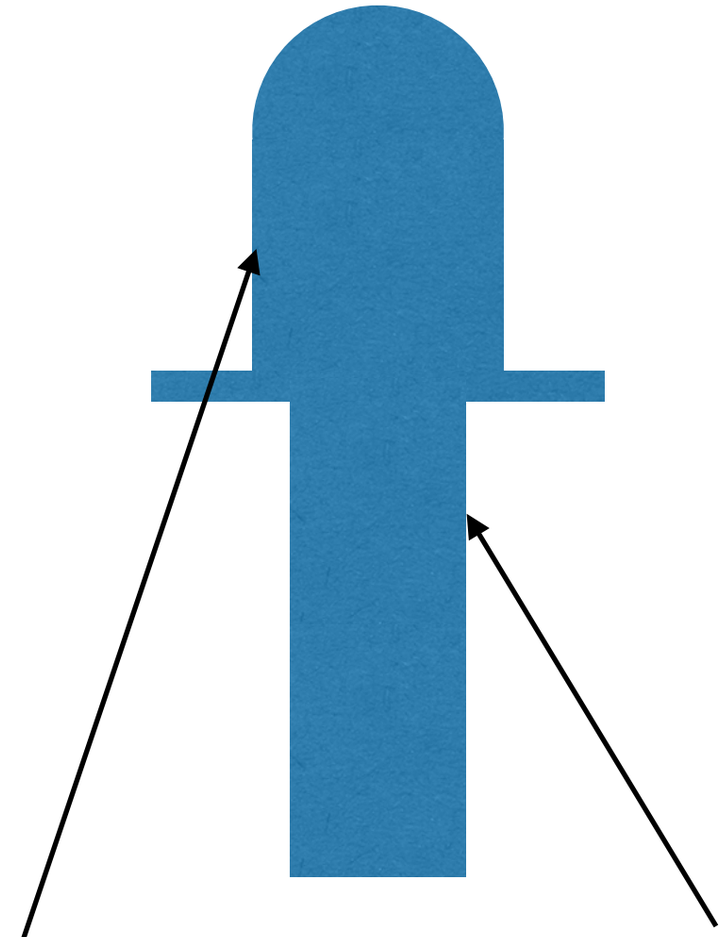
From http://www.brailleauthority.org/sizespacingofbraille/sizespacingofbraille.pdf

Memory Wire Specifications

- Maximum Deformation Ratio is 3-5%
- For 075 memory wire (the initial wire used but rejected because of how thin it is)
    - Wire Diameter = 75μm
    - Minimum Bend Radius = 3.75mm
    - Linear Resistance = 200 Ω/m
    - Recommended Current = 100 mA
    - Recommended Power = 2.0 W/m
    - Needed length of wire to pull down pin = 40 mm; the pin needs to move down 0.7 mm and the wire contracts 5% so 0.7 / 5% = 20 mm but its going through both ends of the needle hole and we're using linear resistance so we need to multiply it by two.
    - Using Ohm's law we can find out how much voltage is needed.

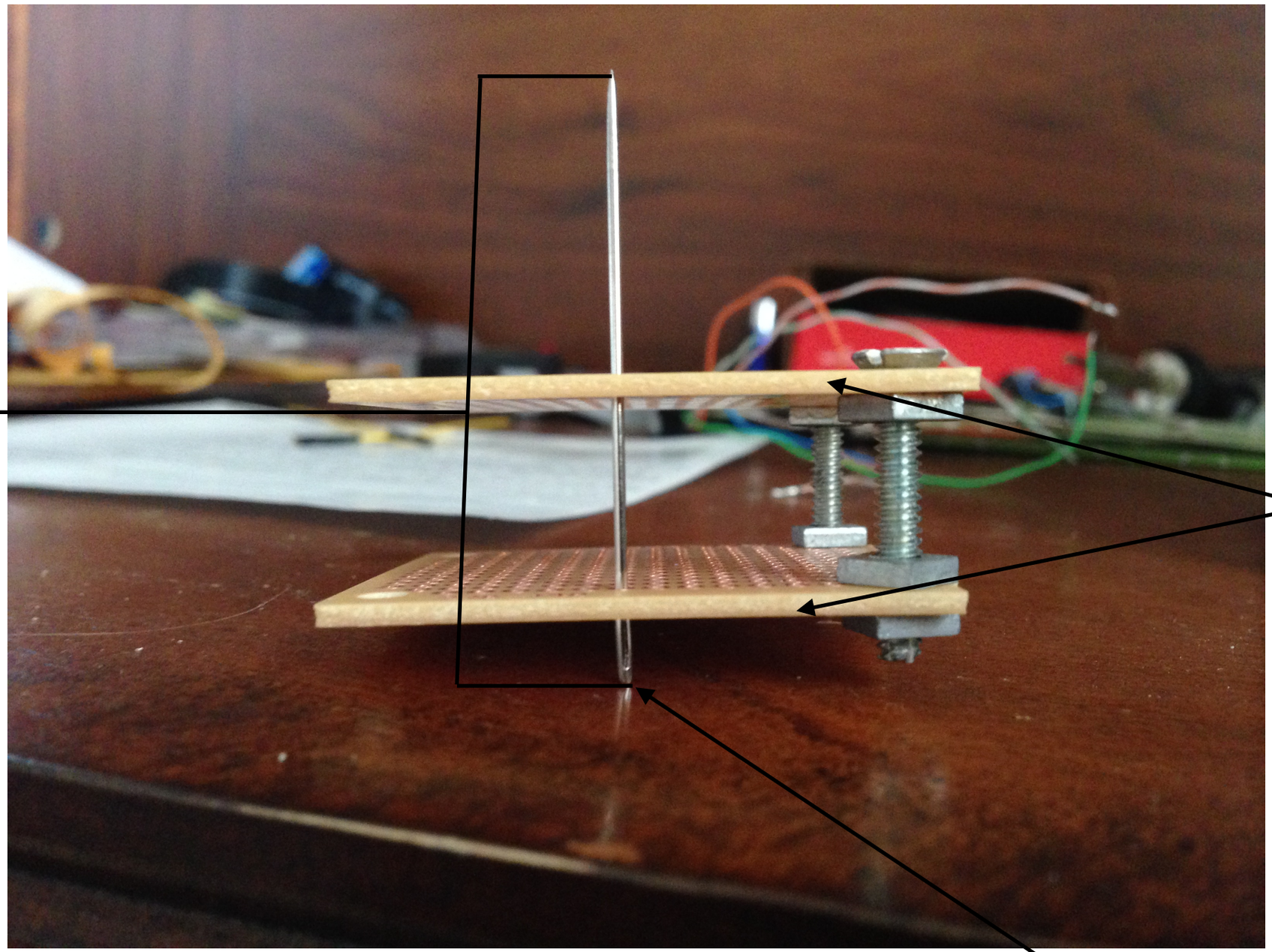
Initial Design





Different thickness because of different measurements need for spring diameter and for Braille dot diameter

First General Design

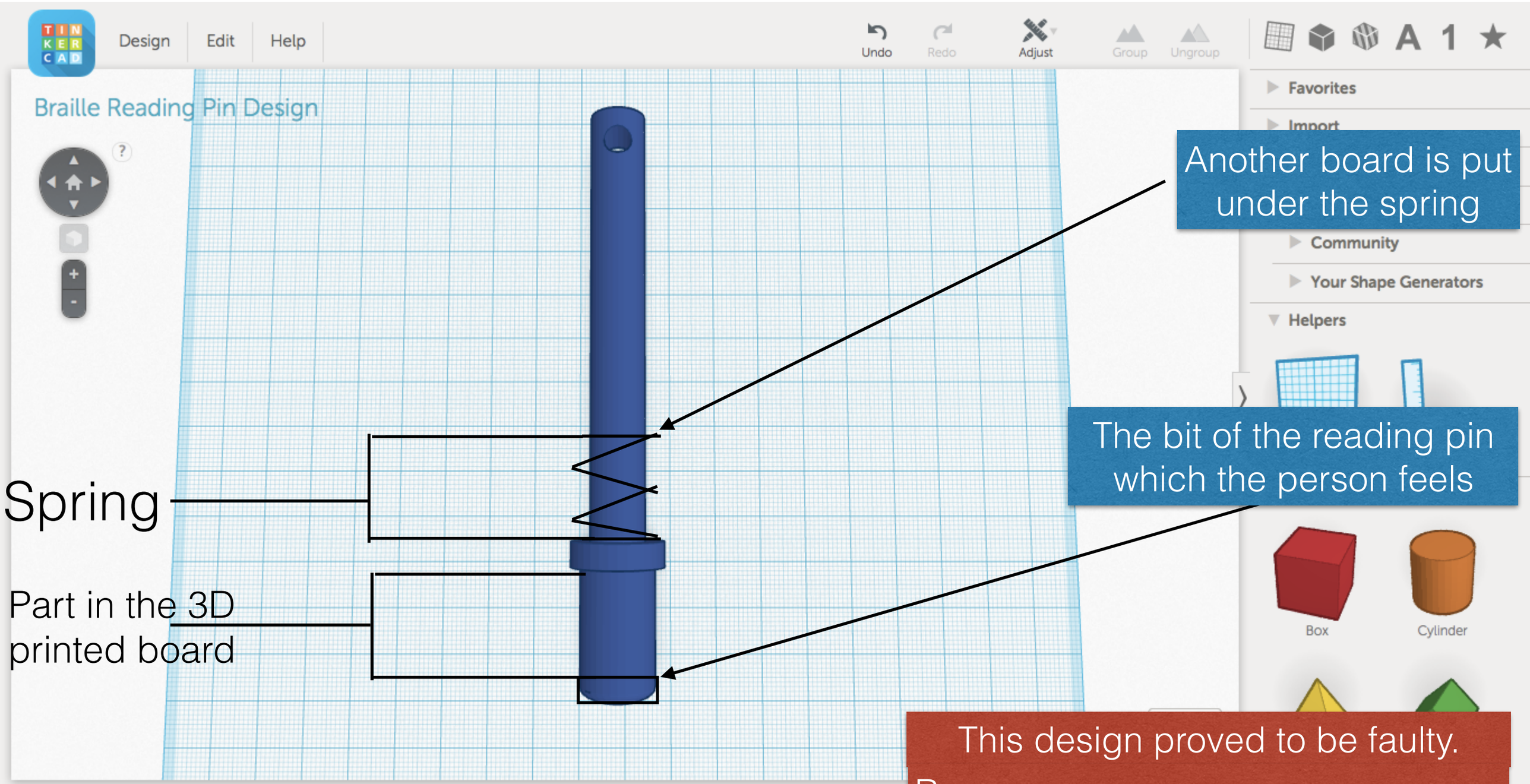


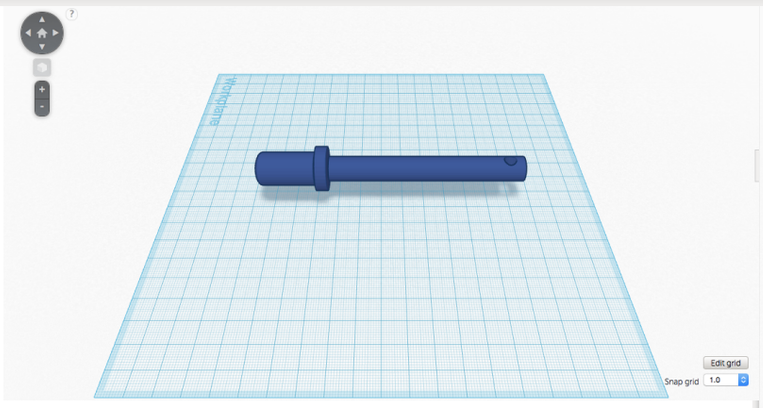
Replacement Pin

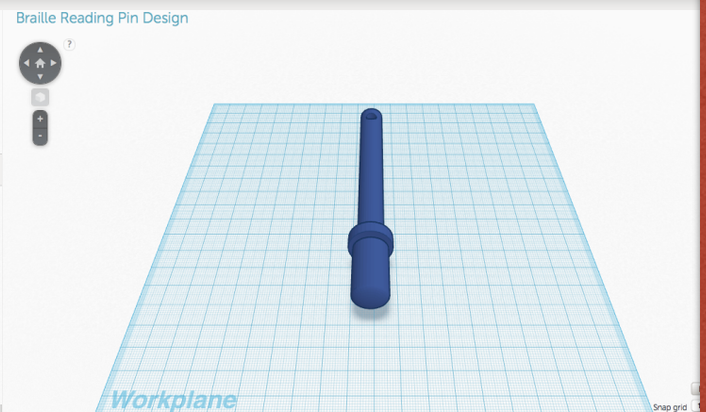
Dual Printed Circuit Board

Hole in “pin” in which memory wire will be put through

Initial Reading Pin Design





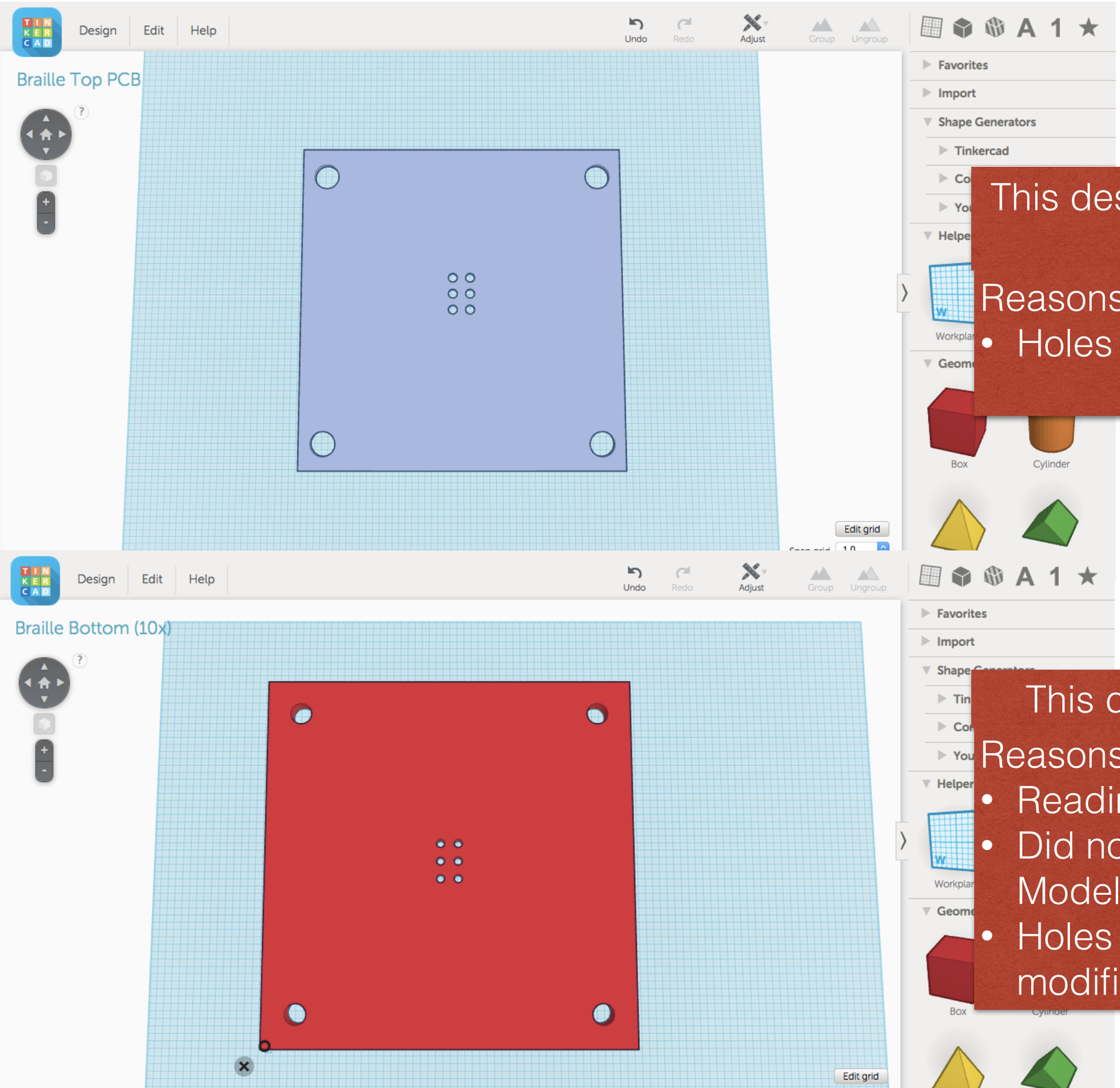


This design proved to be faulty.

Reasons

- Very flimsy
- Too big to fit in bottom board
- Too small for spring to fit neatly
- On some pins, the bottom part including the hole was nonexistent

3D Printed Board Models



This design proved to work but was still changed.

Reasons

- Holes for screws weren't workable

This design proved to be faulty.

Reasons

- Reading pin holes were too small
- Did not line up with the Top Board Model
- Holes for screws need to be modified

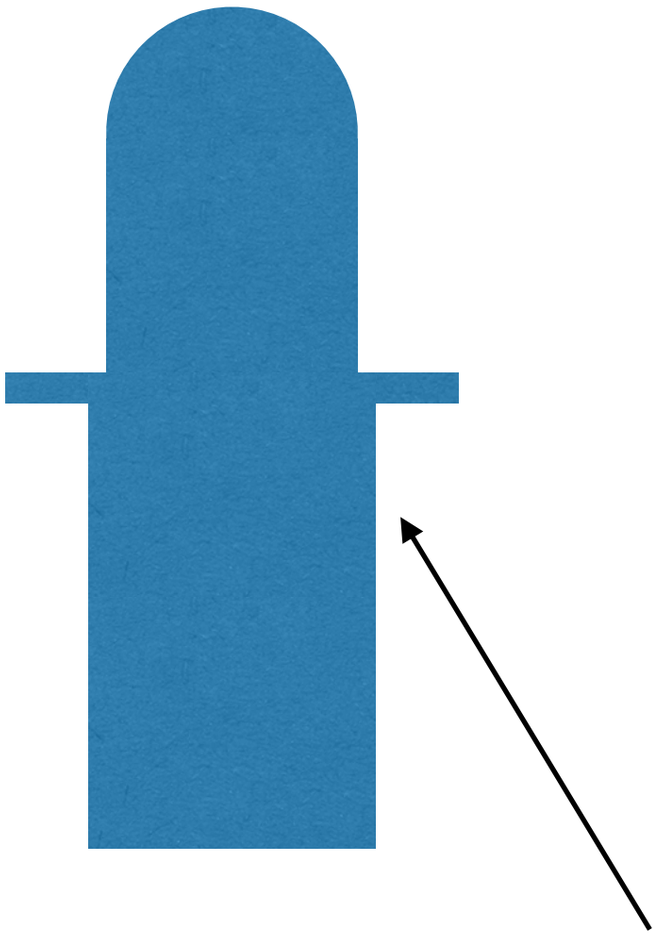
Second General Design/Prototype

Using three dual-standard PCBs, I managed to make a prototype that theoretically would work besides for the fact that the memory wire was too thin and couldn't pull the spring and needle down.

Spring

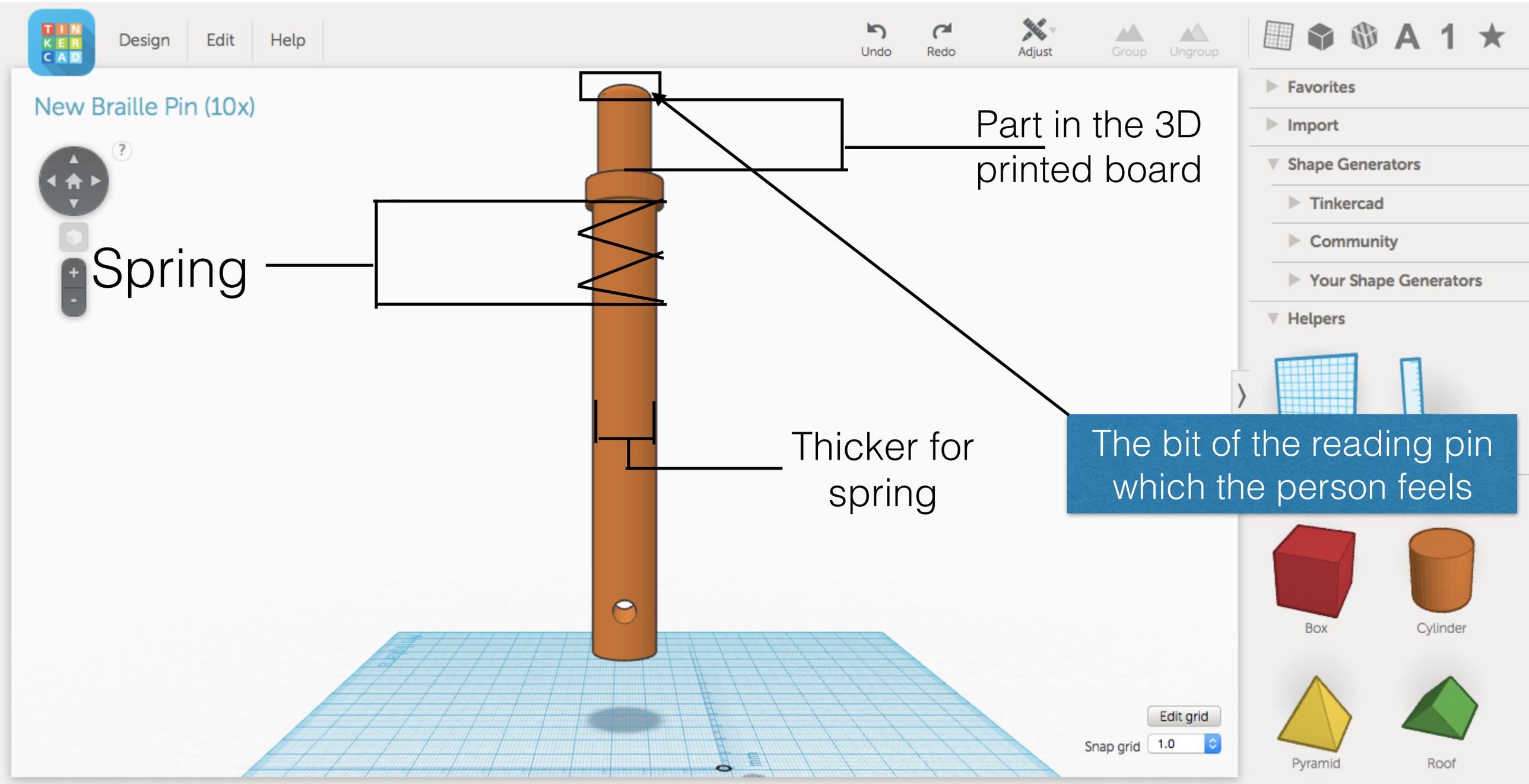
Memory Wire

Final Design

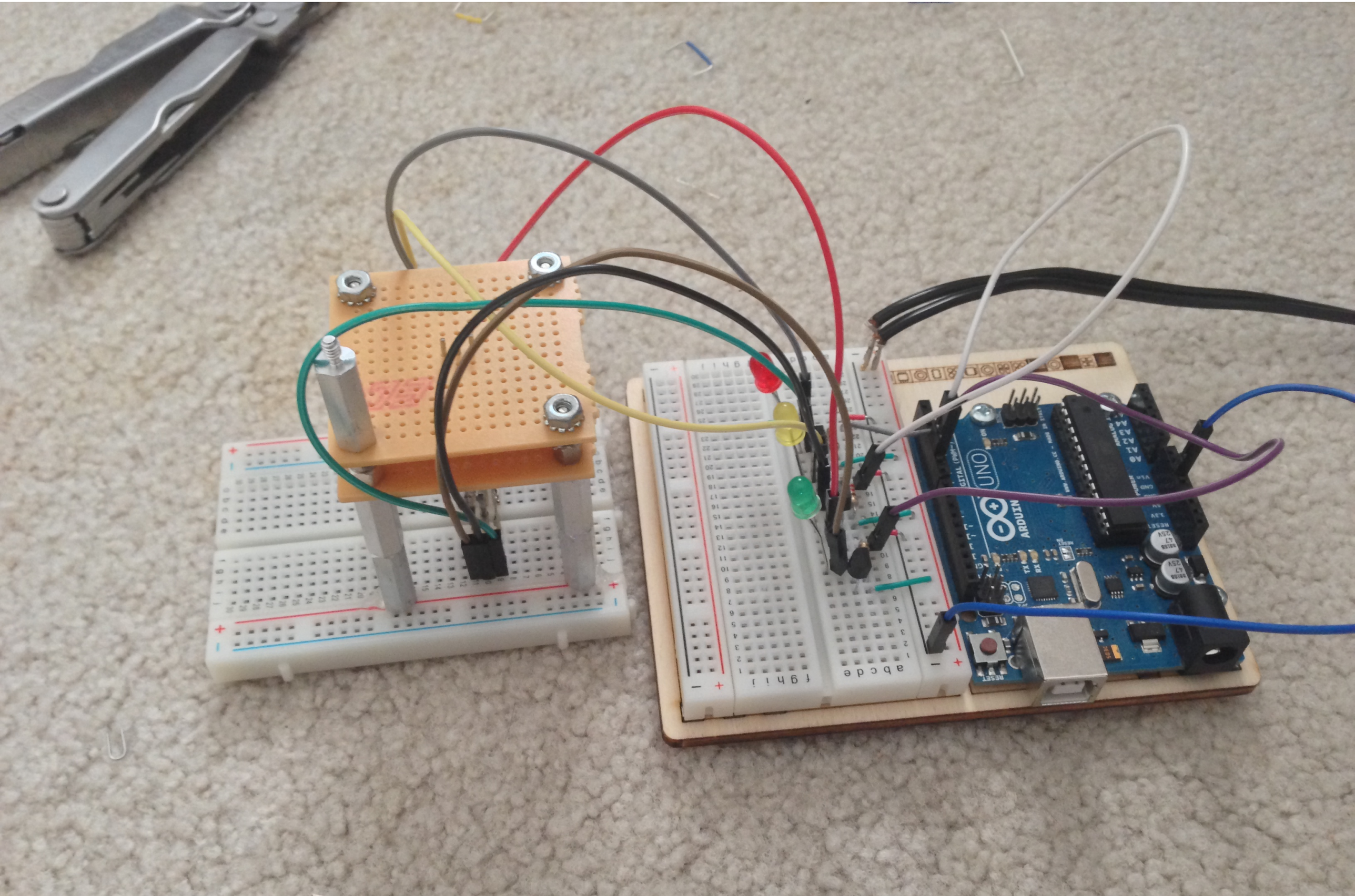


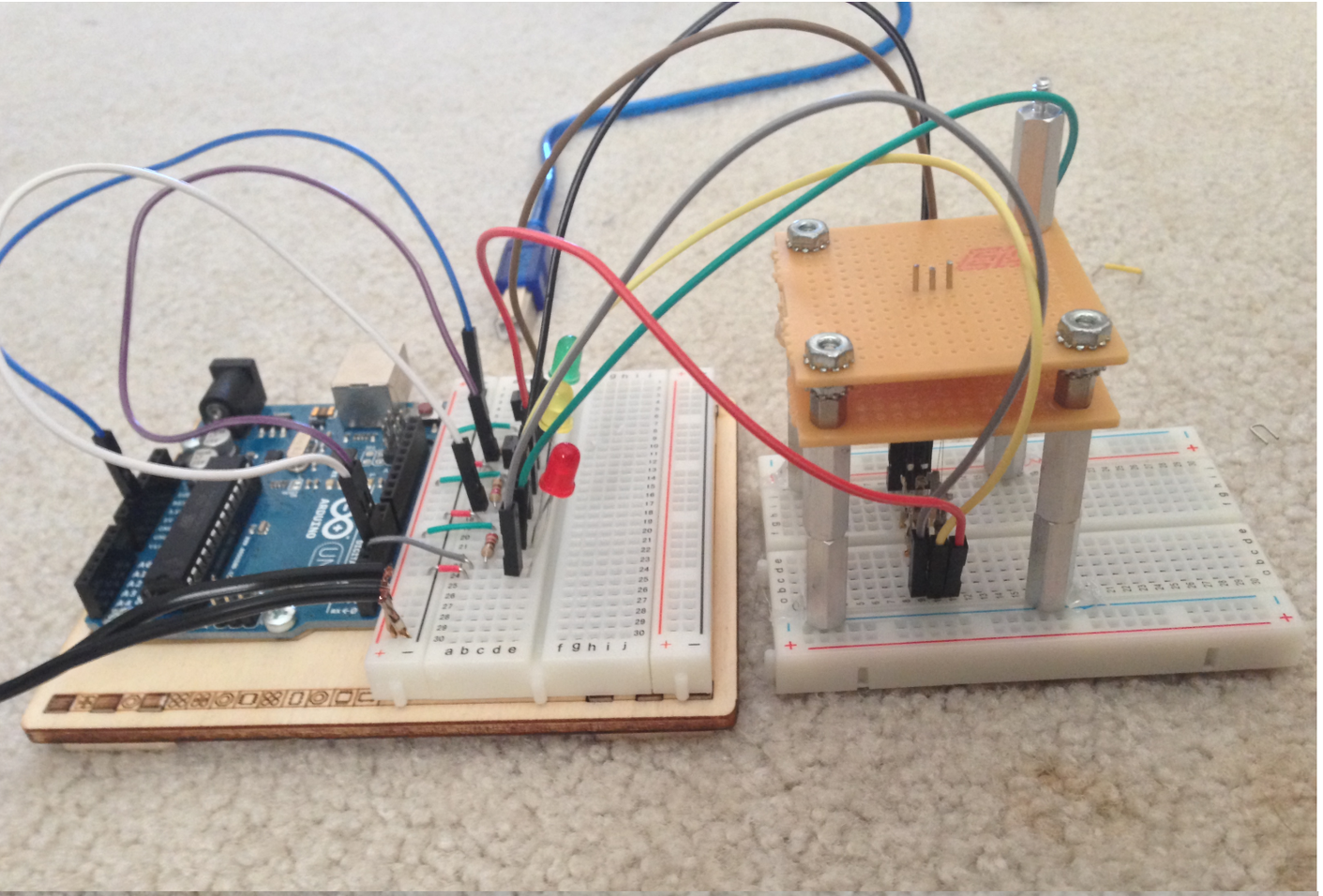
Different thickness because of bad spring measurements in the beginning.

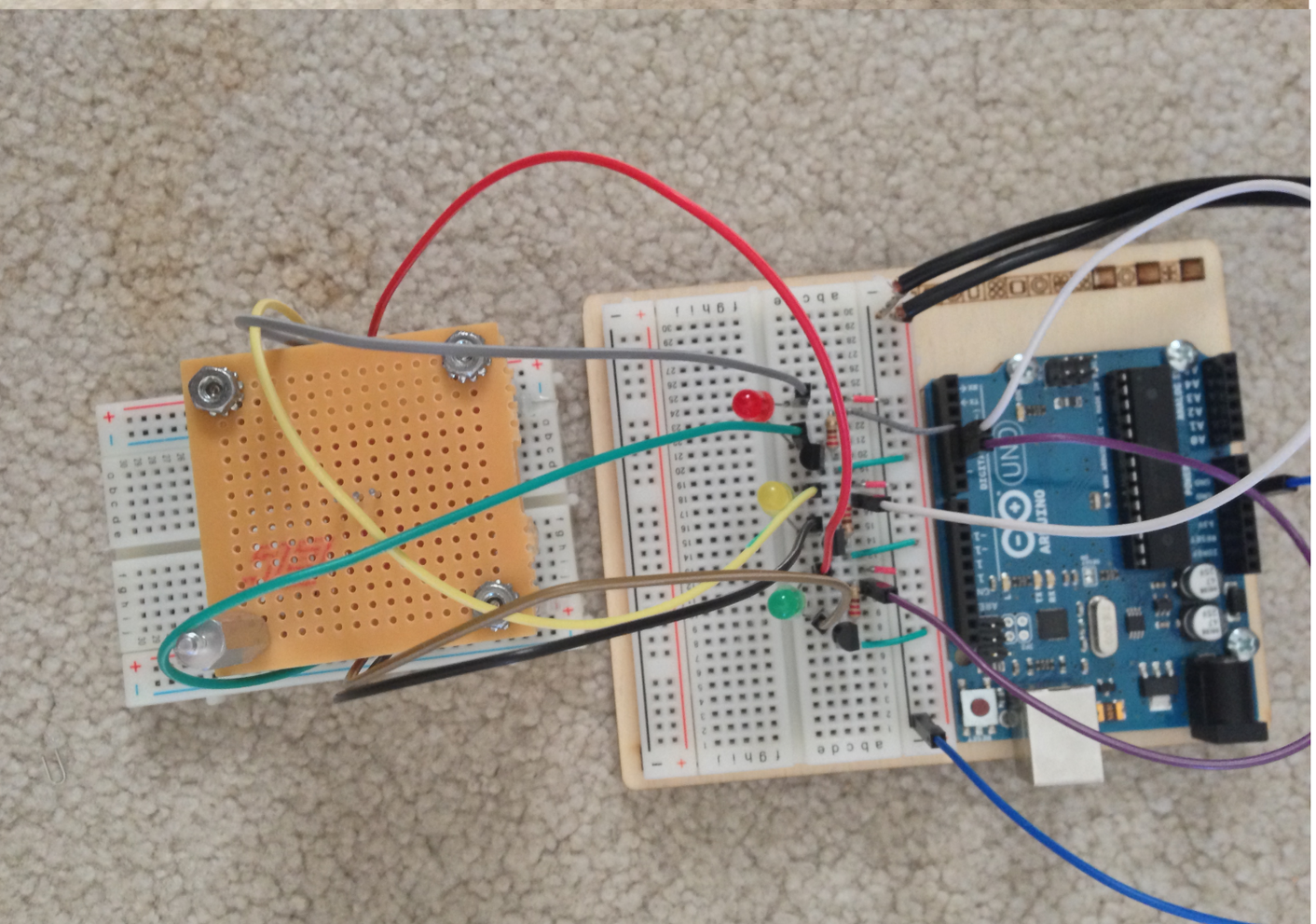
Second Reading Pin Design



Third Prototype with Arduino and working memory wire



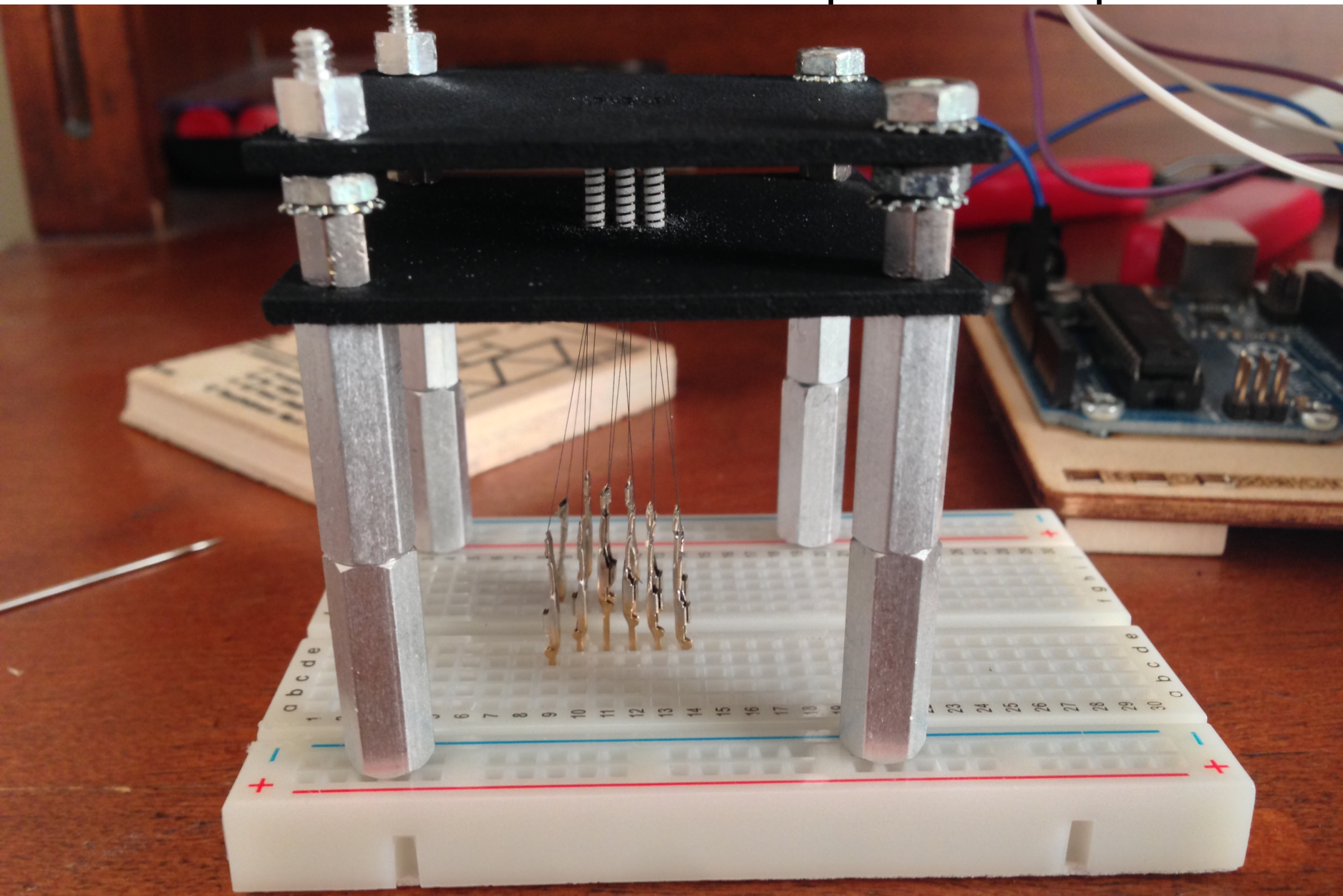




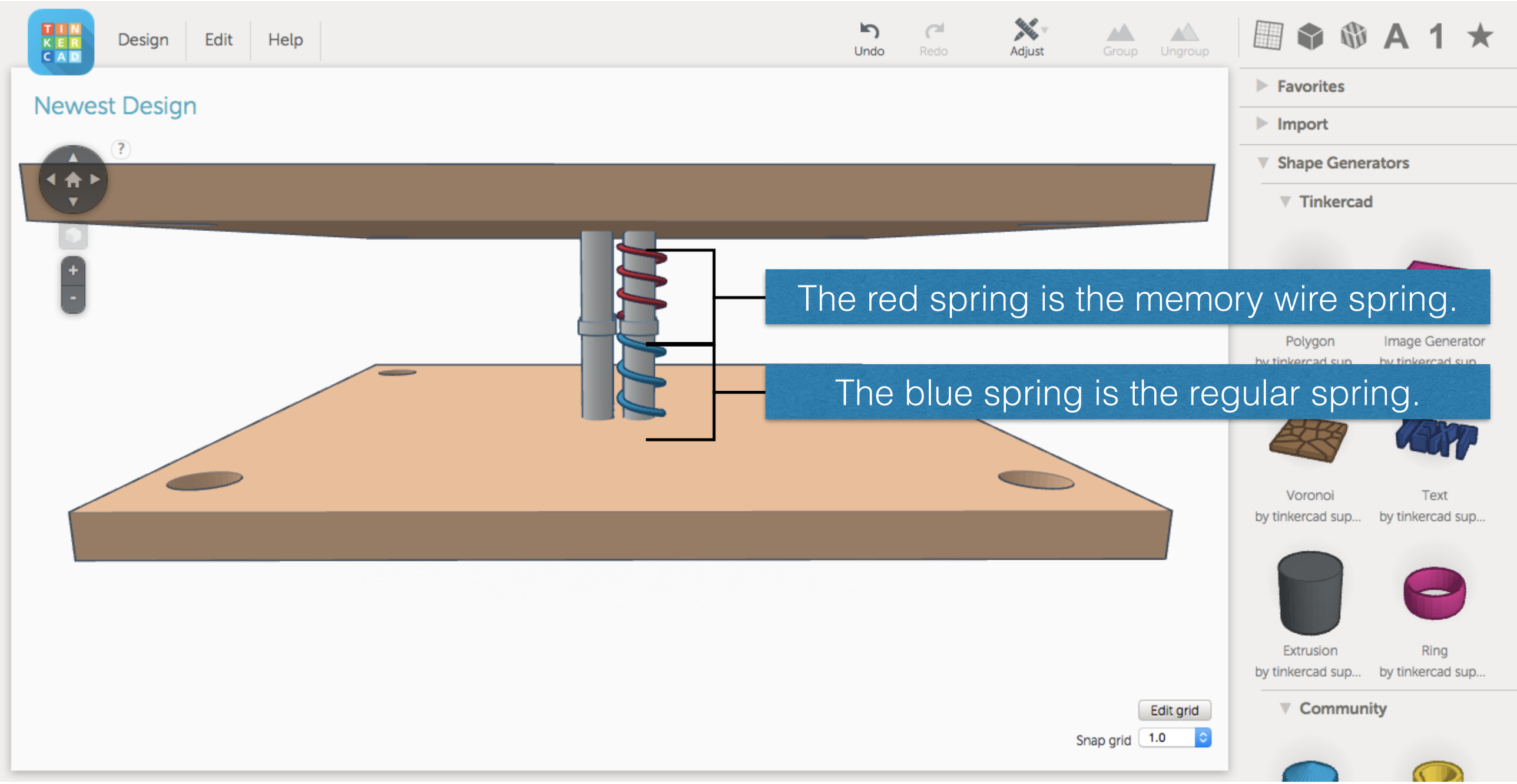
- Using the second prototype's design I built a similar prototype with thicker memory wire and Dupont pins which can be easily put in the breadboard.

- There are three pins and I can control all of them separately using different transistors and the LEDs are used to troubleshoot in case something goes wrong (like it usually does).

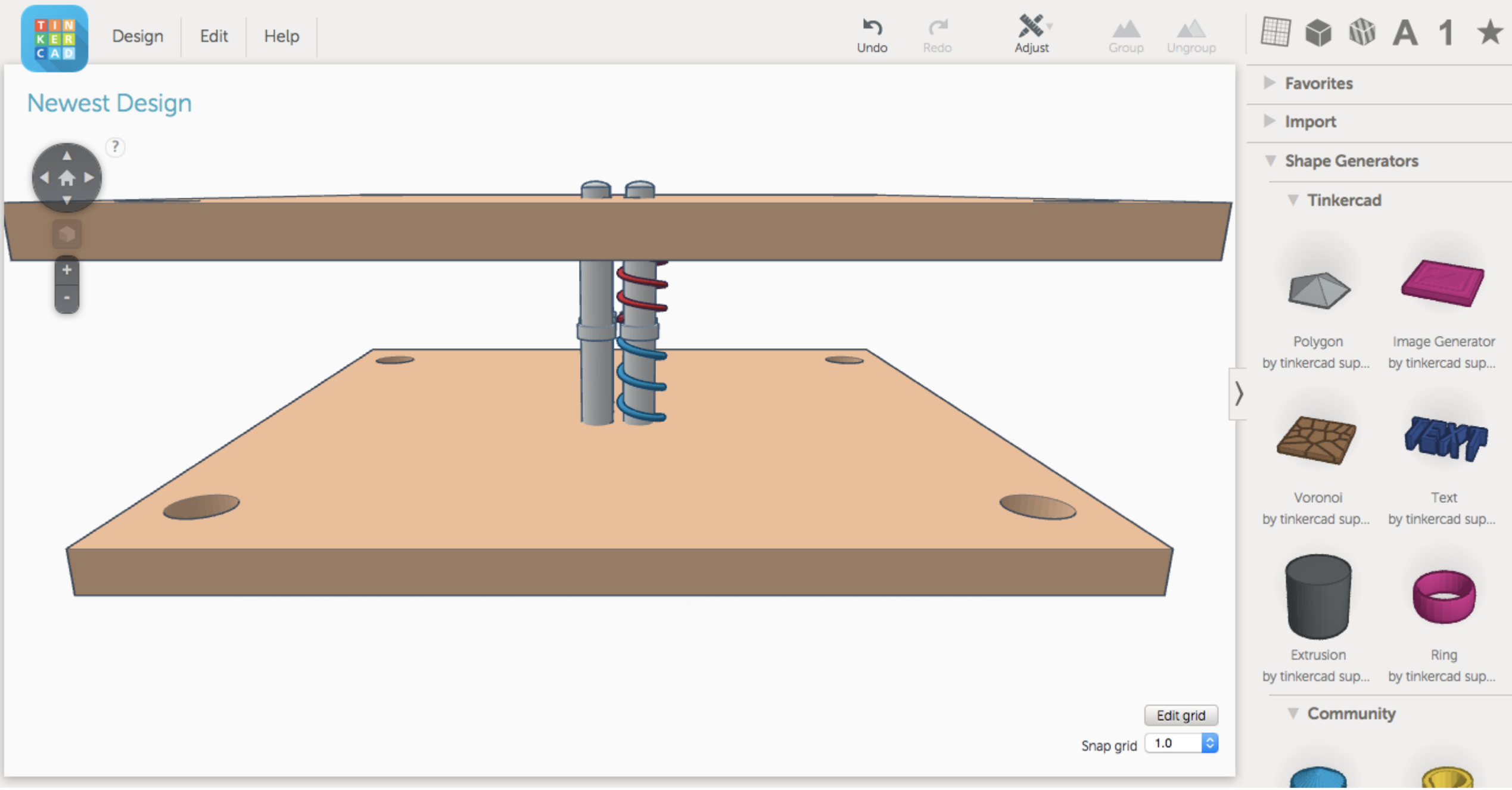
Final Version with 3D-printed parts



Newest Best/ Most Practical Design

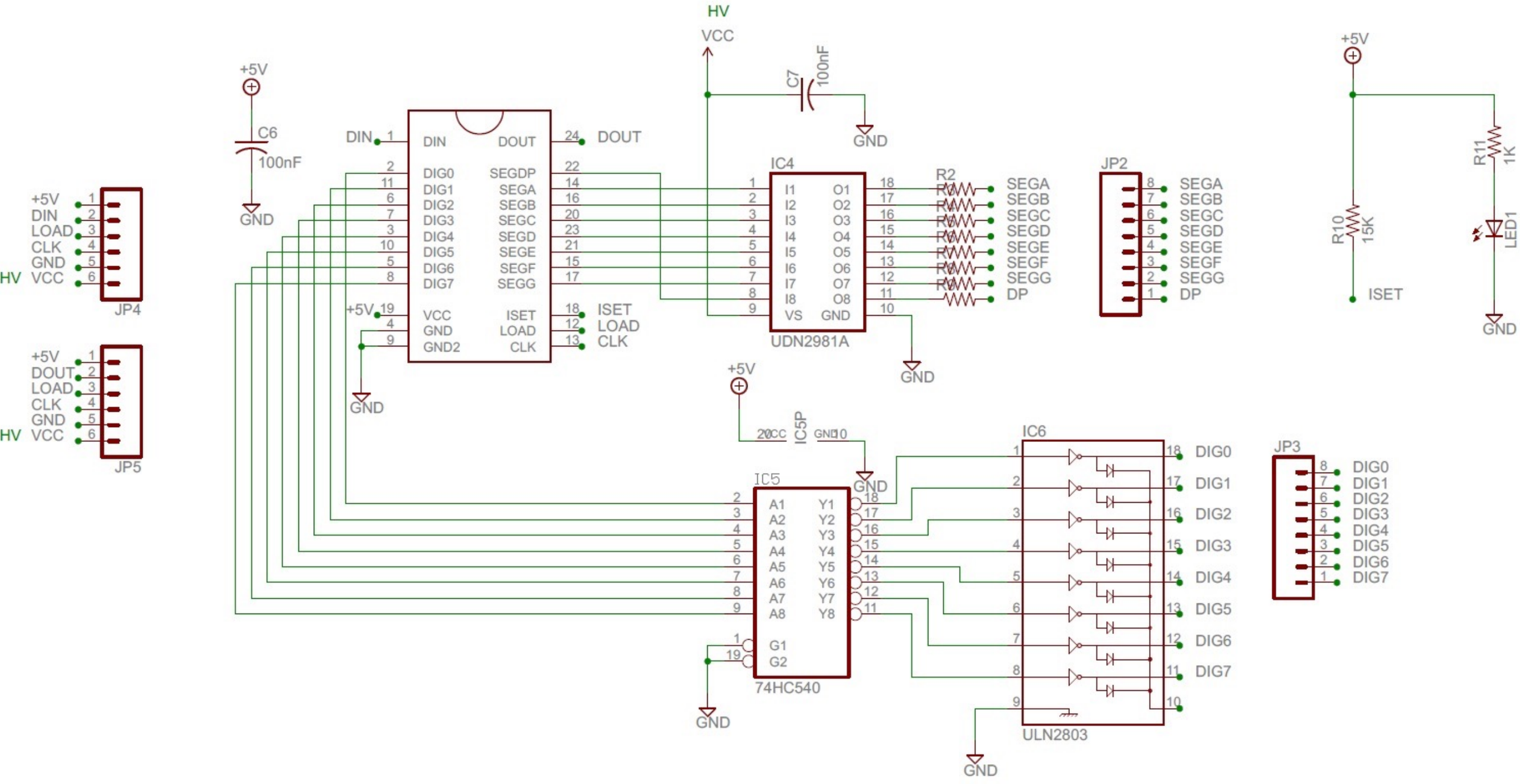


The top of the memory wire spring would be attached to the top of the 3D-printed board and the regular spring would follow a similar suit except be attached to the bottom board. This way the springs can push against each other without moving. The memory wire spring would push harder against the regular spring causing the pin to be pushed down by default. When heat is applied the memory wire would contract causing the regular spring to win the push fight and making the pin go up.



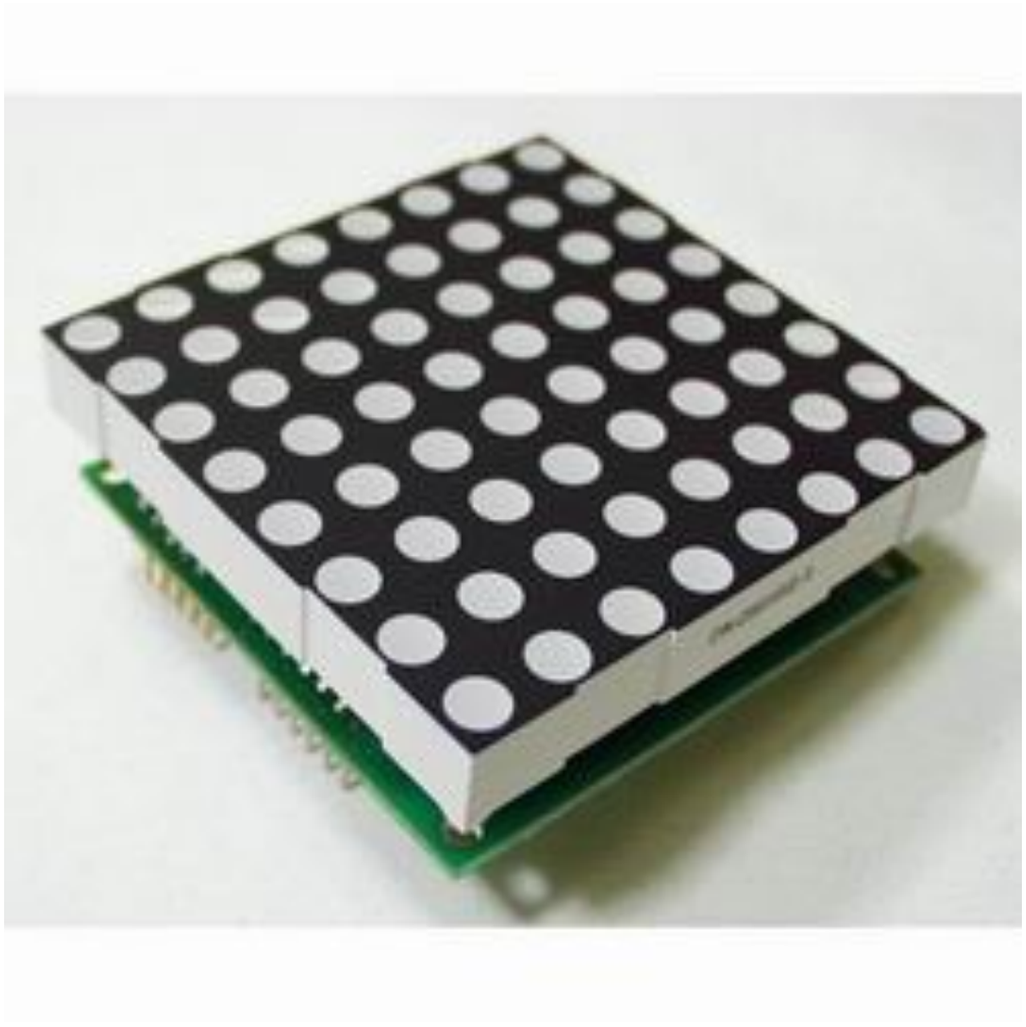
This design would work well potentially if the supplier for the memory wire sold springs of similar size as the pins. Unfortunately, they sell springs but not of the right size. Custom size springs can be ordered but it is too expensive for a small number.

Future Work: Easy Control over many pins



These are schematics that allow the setup of MAX7219 with high power. The chip alone cannot output high currents but with another chip, UDN2981A, a 8-channel source driver it can output high current to the pins.

Future Work: LED Array



Using a similar design, I could remove the LEDs from this LED array and connect the wire from the reading pins for my Braille project. This would simplify installation and be much neater. The chip used to control the LED array, MAX7219, can control the 8x8 LEDs but it cannot work directly with the Braille display because the maximum current is too low.

Future Work

I have made a single character Braille display that can be controlled with an Arduino. In the future, I hope to make a bigger display with more characters. The average amount of characters on a one-sided page is 3,000 characters.

Conclusion

I have made several prototypes of a Braille display; most of which did not work. The final version with the 3D-printed Braille pins and boards works but still is not optimal and isn't practical. It may be cheaper and better than most on the market but it is impractical and not ready for commercialization.

Citations

- http://www.brailleauthority.org/sizespacingofbraille/sizespacingofbraille.pdf
- http://www.freepatentsonline.com/6700553.html
- http://spie.org/x37076.xml?ArticleID=x37076
- http://www.techfresh.net/braille-e-book-concept/
- http://www.yankodesign.com/2009/04/17/braille-e-book/
- http://www.musclewires.com/MWFlexinol.php
- http://embedded-lab.com/blog/?p=7628

Acknowledgements

Many thanks go to Bradley James Cooke, Robert Bauer, and Latchesar Ionkov for helping me with the project.