Glowing Viking Rune wayFinder

Created by Erin St Blaine

https://learn.adafruit.com/glowing-viking-rune-artifact

Last updated on 2024-06-03 02:05:31 PM EDT

Table of Contents

Introduction

I joined an all-female barbarian cosplay group. We're called "The Horde".

My character is a priestess, and of course I just had to have some light-up elements in my costume. At the same time I also really want to keep the whole costume within the Horde's costume guidelines, which means everything needs to look ancient and weathered -- rather like I stole it from a defeated warrior after an epic battle.

This project attempts to melt the newest fanciest tech I can get my hands on with the most ancient stories and legends I've ever heard.

Electronics Materials

1 x Pro Trinket Pro Trinket 5v 16MHz	https://www.adafruit.com/product/2000
1 x LiPoly Backpack Charger Battery Charger	https://www.adafruit.com/product/2124
1 x Neopixel Skinny 144/m Skinny Neopixels	https://www.adafruit.com/product/1506
1 x On/Off Switch Slide Switch	https://www.adafruit.com/product/805
1 x Momentary Switch 6mm Tactile Switch Button	https://www.adafruit.com/product/1119
1 x LED Sequin Warm White LED Sequin	https://www.adafruit.com/product/1758
1 x Battery LiPoly 150mAh Battery	https://www.adafruit.com/product/1317

For the case:

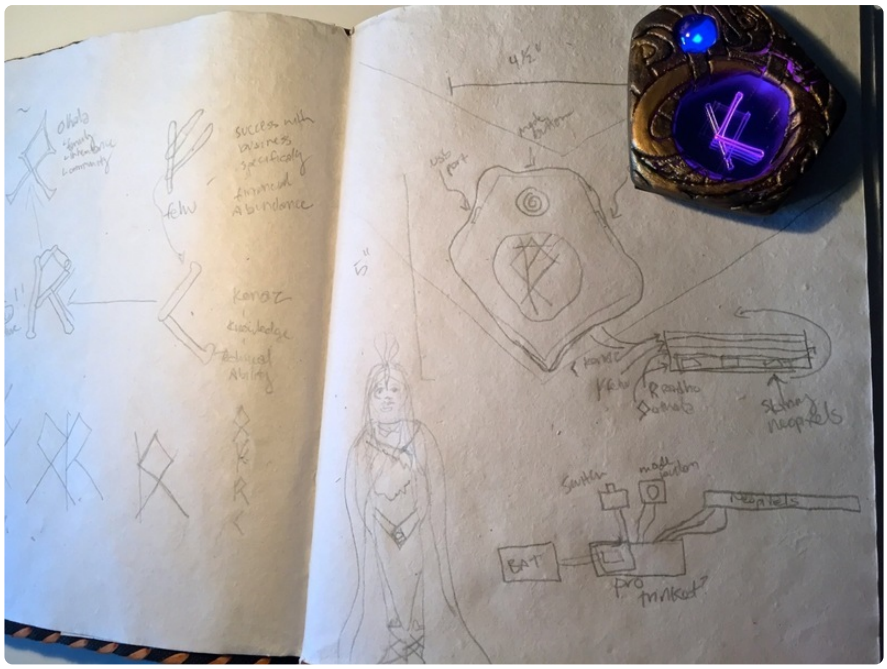
- 3d printer
- Acetone
- Metallic Acrylic paints
- Hot glue

- Black Gaffer's tape
- Glass gem for the button

Design & Planning

Waystones dot the fantasy Barbarian landscape -- standing stones are happened upon by weary travelers, and woe bedite any who fall asleep beneath a Waystone on a full moon night. History tells us the Ways once opened to secret passages throughout the kingdoms; although most folk these days have forgotten their use and dismiss them fairy stories.

But if a traveler has a Key to such a stone, and knows some of the secrets of its use, untold adventures await just beyond the edge of dreaming.

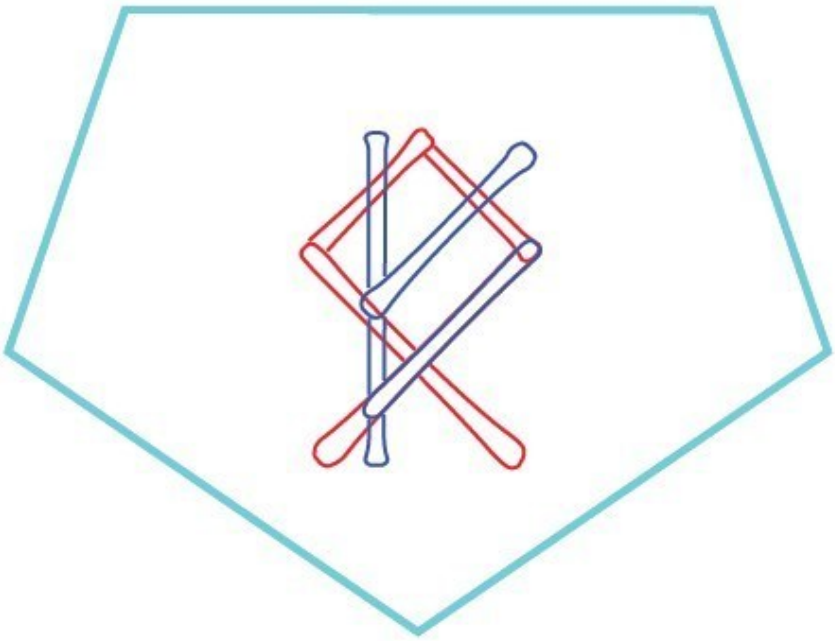


Four viking runes are layered on laser-cut acrylic, suspended inside a darkened box. Upon pressing the button, the runes will fade through lights and colors and stop on a random single or double rune, to give a fortune a-la the mysterious and timeless Magic 8 Ball.

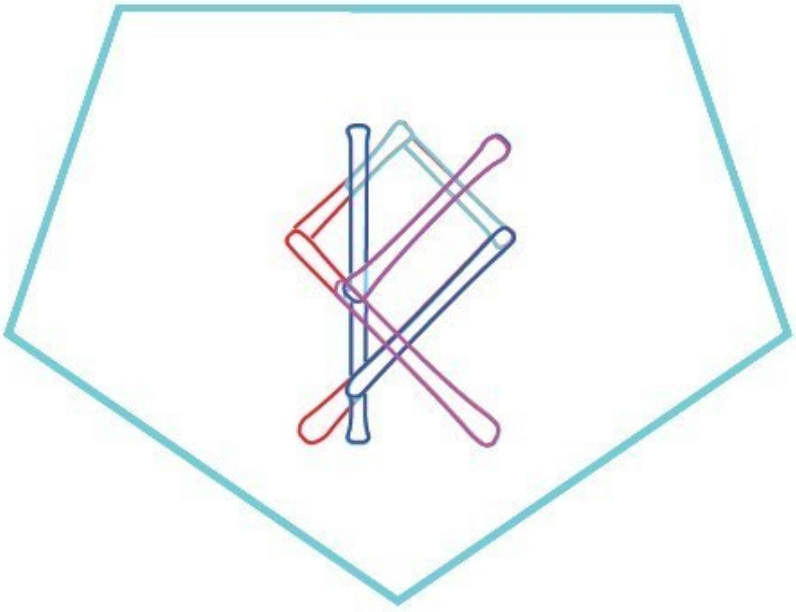
I'll use 3d printed plastic for the case because I want it light and detailed and translucent (and besides which, I'm just not that skilled a metal worker). Making it look like metal is going to take a bit of work, but we'll see how close we can get.

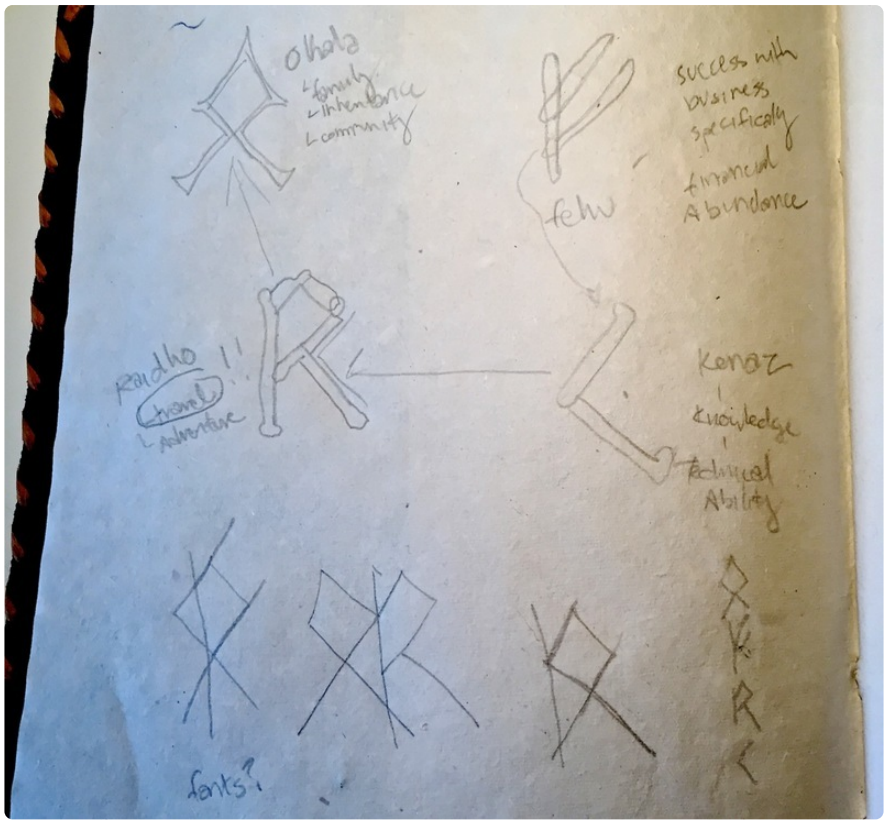
The runes will be cut on four thin sheets of acrylic which will then be stacked up and lit from the side, one or two at a time. This will create depth and interest in the device since the runes should effectively look like they're floating in space.. at least, that's the goal.

First step was to decide on my runes. I started with one I got from a fortune teller -- Othala, meaning family ties and inheritance -- and then did a little Rune Internet Research (https://adafru.it/v6e). I added Fehu, another one I found on the internet that resonated with me (it means business success) and also looked good when stacked up with Othala. I played around with them for a while in photoshop until I found a configuration I liked.

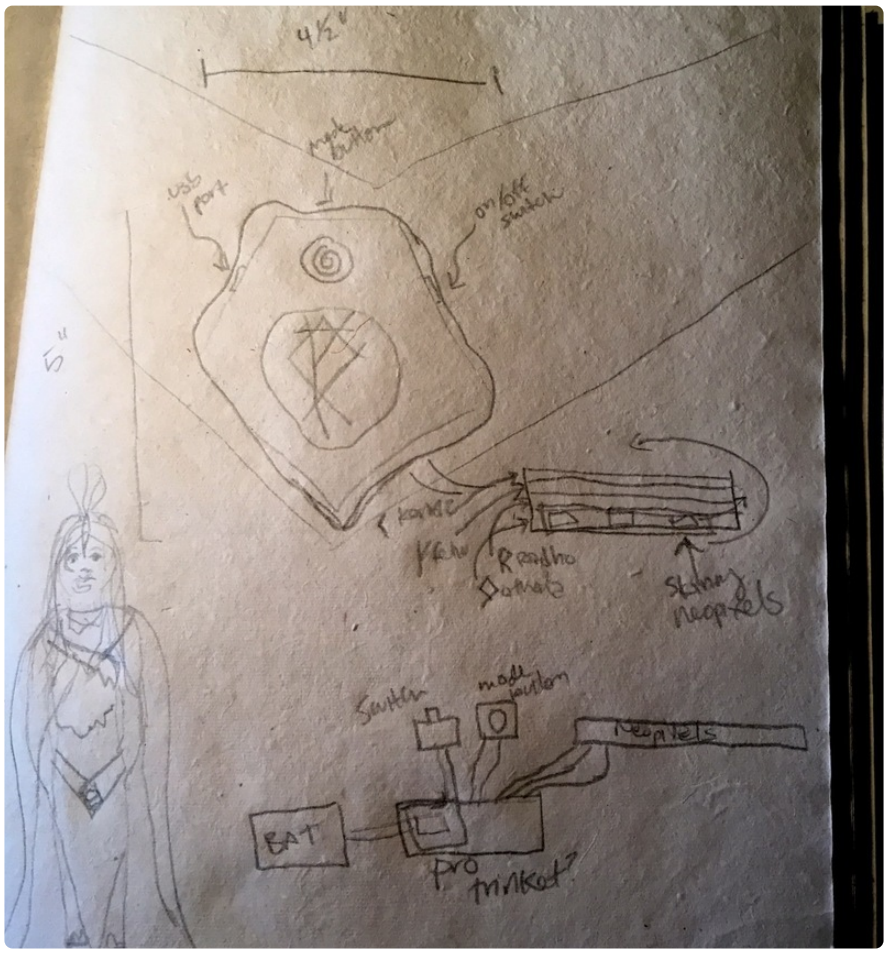


When I stacked them on top of each other and looked closely, a bunch more hidden runes appeared (just like magic!). I did a little more research and ended up choosing two others, Raidho and Kenaz, giving me runes for travel and for technical ability. Four feels like the right number but i like that there are more hidden in here too, if you know how to look.



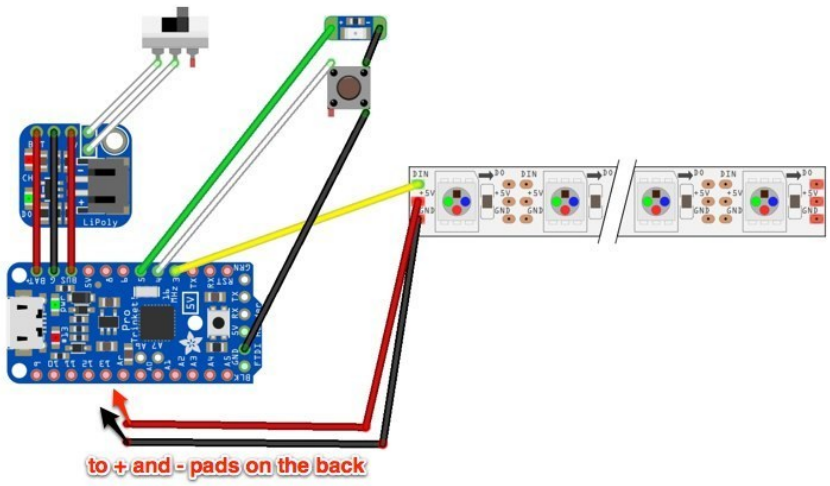


Once I had the runes, I did a couple quick sketches to imagine layout and the general shape and size for my piece.



The case design and electronics layout developed hand-in-hand, and several ideas were tried and rejected before I settled on the Pro Trinket and the final sizing and layout of the box.

Wiring Diagram



The Pro Trinket Lion/Lipoly Backpack will be soldered on to the Pro Trinket using the included headers.

Add the on/off switch to the switch pads on the Backpack, remembering to cut the trace between the two pads to enable the switch.

The button will be wired to pin 4 and G, and the LED sequin will be wired to pin 5 and G -- it's convenient to use the other leg of the button since G pins are a bit in short supply.

Connect the neopixel strip data to pin 3 (remember to connect from the "in" end) and connect the 5v and G pins to the + and - pads on the back of the Pro Trinket.

Plug the battery into the charger and you'll be good to go! You'll be able to charge the battery without opening the case via the USB port on the Pro Trinket.

Code

Before you Start

If this is your first time using Pro Trinket, pause for a sec and head over to the Pro Trinket Intro guide (https://adafru.it/iPe). That will walk you through getting your Arduino IDE set up and the Adafruit boards installed. Once you can upload sample code to the Pro Trinket and everything appears to be working, come on back here.

FastLED Library

You will also need to install the **FastLED** library in Arduino (`Sketch > Include Library > Manage Libraries...`)

Upload Code

This code is based on Mark Miller's sample beat8 code for the FastLED library. It creates a moving pixel with a comet tail that moves in a sinwave pattern around the neopixel strip.

When I press the button, the lights cycle through all the runes for a few seconds and then stop in a random spot, lighting up one (or sometimes two) of the layers of acrylic and highlighting a specific rune or "bound" rune combination (if it lands on a corner).

The lit pixels can be controlled for speed, brightness, hue, and fade rate (via the fading "comet tail" effect). The sinwave code "beatsin8" effectively mixes all these

variables so that each button press gives me a totally unique experience -- one press, the pixels might cycle bright and fast, the next they cycle dim and slow, change colors quickly or subtly, or any of a vast number of combinations.

This makes for a feeling that the device is "listening" to the user and giving each button press due consideration before deciding on a message to relay.

```
// Viking Rune Artifact
// by Erin St. Blaine
// for Adafruit Industries
// Based on code by Marc Miller, March 2017

//  Full tutorial at https://learn.adafruit.com/glowing-viking-rune-artifact/
introduction
//  Pro Trinket: https://www.adafruit.com/products/2000
//  Skinny Neopixels: https://www.adafruit.com/products/2970
//*****************************************************************

#include "FastLED.h"
#define LED_TYPE WS2812
#define LED_PIN     3
#define BUTTON_PIN  4
#define BUTTON_LIGHT_PIN 5
#define NUM_LEDS    24
#define COLOR_ORDER GRB
CRGB leds[NUM_LEDS];

uint8_t moveSpeed;// Random number for movement speed - higher number cycles faster
uint8_t fadeRate;  // Use lower value to give a fading tail.
const uint16_t startDelay = 100;  // Number of milliseconds before re-start
int analogSeedPin = 9;  // Any un-used analog pin.  Used for generating seed for
random16.
int analogSeedPinB = 10;  // Any un-used analog pin.  Used for generating seed for
random16.
bool oldState = HIGH;
static boolean frozen = true;  // [Initially set true!]
uint8_t gHue = 0;  // Used to cycle through rainbow.

uint8_t BRIGHTNESS = 200;
unsigned long timeDelay = startDelay;
unsigned long freezeTime = 10000000;  // Just something really large to prevent
triggering on startup


//-------------------------------------------------------------
void setup()
{
  Serial.begin(115200);  // Allows serial monitor output (check baud rate)
  //delay(3000);  // 3 second delay
  FastLED.addLeds&lt;LED_TYPE, LED_PIN, COLOR_ORDER&gt;(leds,
NUM_LEDS).setCorrection(TypicalLEDStrip);
  FastLED.setBrightness(BRIGHTNESS);
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  pinMode(BUTTON_LIGHT_PIN, OUTPUT);
  digitalWrite(BUTTON_LIGHT_PIN, HIGH);       // turn on pullup resistors// set pin
to output
}


//-------------------------------------------------------------
void loop()
{
  EVERY_N_MILLISECONDS(random(1,8)) { gHue++;}  // Cycle through the rainbow
randomly
```

```
  // Get current button state.
  bool newState = digitalRead(BUTTON_PIN);
// Check if state changed from high to low (button press).
  if (newState == LOW &amp;amp;&amp;amp; oldState == HIGH) {
    // Short delay to debounce button.
    delay(20);
    // Check if button is still low after debounce.
    newState = digitalRead(BUTTON_PIN);
    if (newState == LOW) {

     frozen=false;  //unfreeze and start animation

     }
    }


  // Set the last button state to the old state.
  oldState = newState;
  timeDelay = startDelay;


  if (frozen) {  // [FROZEN]

    // This if statement delays the start of the moving pixel.
    if (millis() &gt; timeDelay) {
      frozen = true;  // toggle frozen variable
      uint16_t seed = random16_get_seed();
      seed = seed + analogRead(analogSeedPin) + analogRead(analogSeedPinB);  //
create a random seed
      random16_add_entropy(seed);
      seed = random16_get_seed();
        freezeTime = random16(2000,6000);  // set range of cycle times here
        moveSpeed= 5;     //movement speed variable
        BRIGHTNESS = random(200,255);  //randomize brightness
        fadeRate = 20;  // change this to change tail length
      Serial.print("freezeTime = "); Serial.print(freezeTime/1000.0);
Serial.println(" seconds.");
      freezeTime = millis() + freezeTime;
    }
  }//end_of_FROZEN


  if (!frozen) {  // [IS NOT FROZEN]
    beat8_tail();  // Subroutine to move the pixel!
  }

  FastLED.show();  // Display the pixels.


  // Check to see if the random freeze time has been reached.
  // If true then stop the animation.
  if (millis() &gt; freezeTime){
    frozen = true;  // toggle frozen variable
    timeDelay = millis() + startDelay;  // set new start time
    freezeTime = millis() + 1000000;  // To prevent re-triggering
  }

}//end main loop


//===============================================================
void beat8_tail()
{

  EVERY_N_MILLISECONDS( 3 ) {
   fadeToBlackBy( leds, NUM_LEDS, fadeRate);  // Fade out pixels.
   }
  uint16_t pos = beatsin8(moveSpeed, 100, 255) % NUM_LEDS;  // modulo the position
```

```
to be within NUM_LEDS
  leds[pos] = CHSV( gHue, 255, BRIGHTNESS);

}
```

To change the way your LEDs act, play with the variables and random numbers in these lines of code:

```
freezeTime = random16(2000,6000);  // set range of cycle times here
       moveSpeed= 5;     //movement speed variable
       BRIGHTNESS = random(200,255);  //randomize brightness
       fadeRate = 20;  // change this to change tail length
```

- `freezeTime` How long the animation runs before freezing. Right now it's set to a random time between 2 and 6 seconds.
- `moveSpeed` make the dot move faster or slower
- `BRIGHTNESS` - I've randomized a bit to add interest and texture
- `fadeRate` this controls the length of the tail, or, how many LEDs light up at once.

Laser Cutting the Acrylic

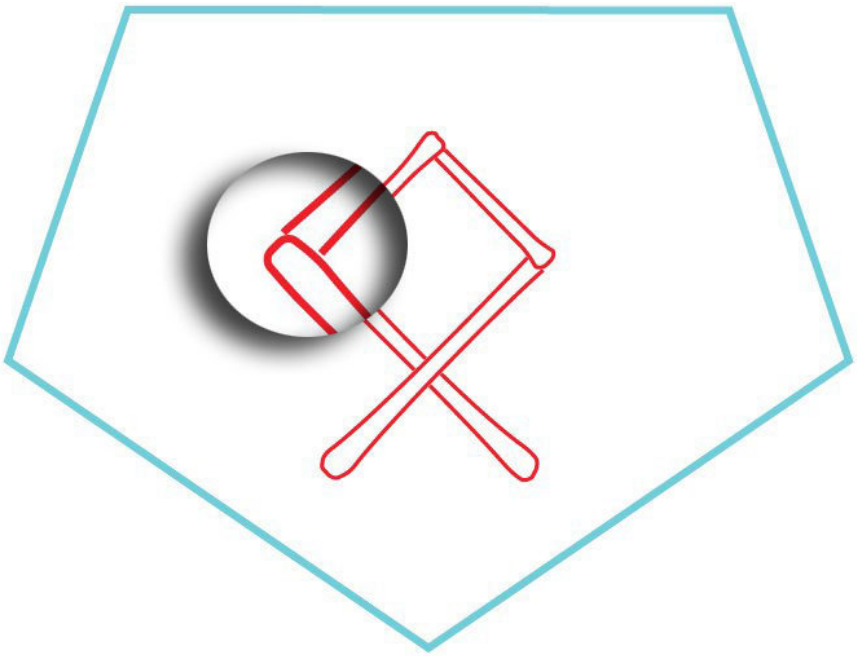
Starting with vector based artwork will work the best when translating to laser cutters and 3d printers. I used Adobe Illustrator to create my design.

Download the file below to get a feel for how it's all put together. This file contains layers for the case as well as the runes and acrylic cut lines.

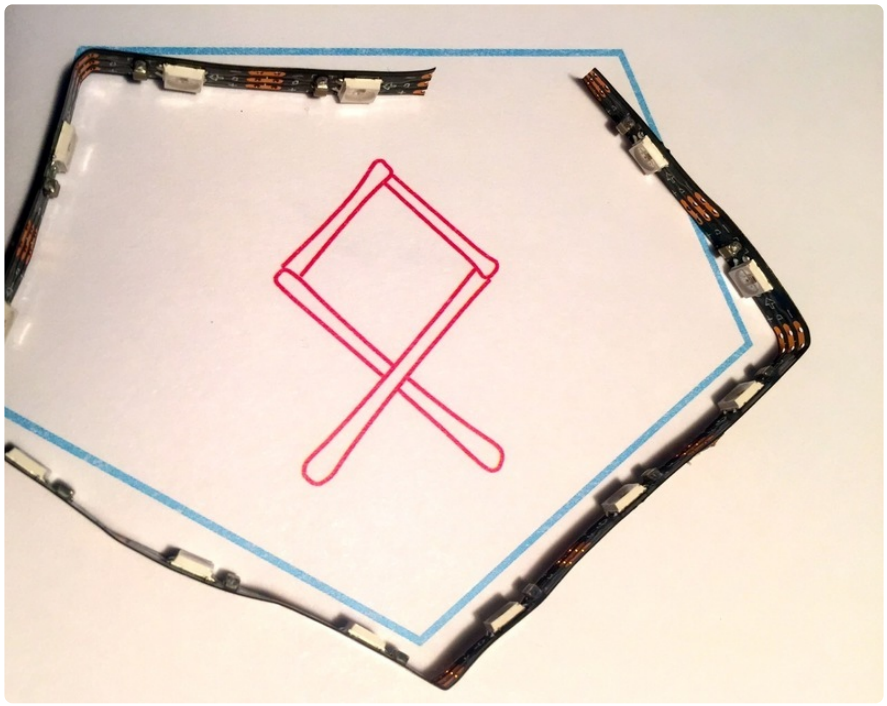
runes_all.ai

https://adafru.it/vd5

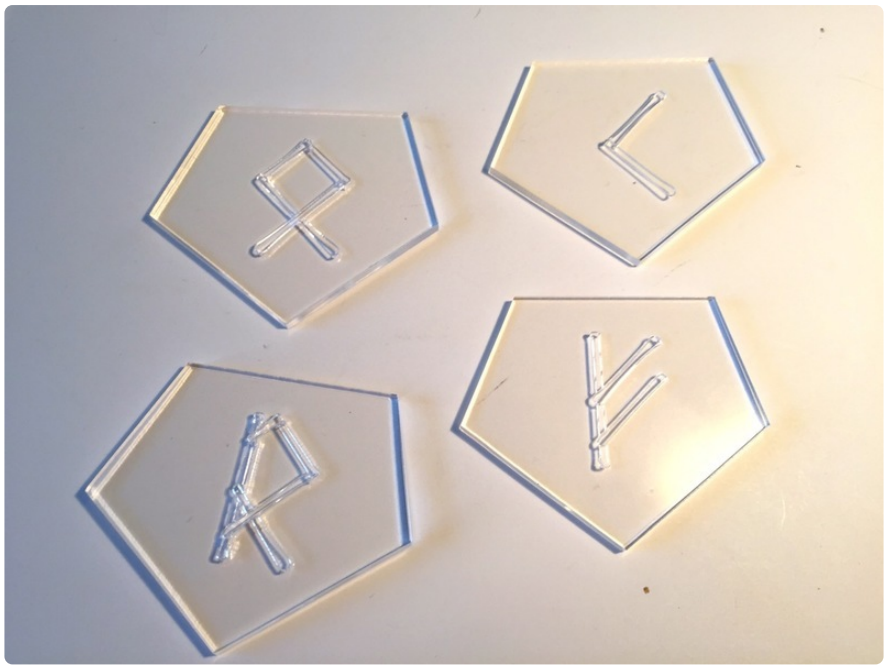
I played around in Adobe Illustrator until I had vector-based images of all 4 runes and a 5-sided outline for the edge of the acrylic. The laser cutter software wants vector artwork only with no fills, just stroke outlines with different colors marking the different depths of the cut. Red here is lines that will be shallowly etched, and blue is for cuts that go all the way through. I made sure the red lines didn't actually cross anywhere since I just wanted the laser to go over each section once.



Print your design out on paper and wrap your neopixel strand around it. Adjust your design so the corners fall neatly in line with the neopixel bend points -- you don't want to end up with a pixel stuck on a corner. My design allows for two pixels on each of the faces on the upper left and right, and three pixels for the two lower faces. (Note: This is a 60/m neopixel strand -- I later used 144/m for the finished project. I also ended up trimming off the ones along the top)



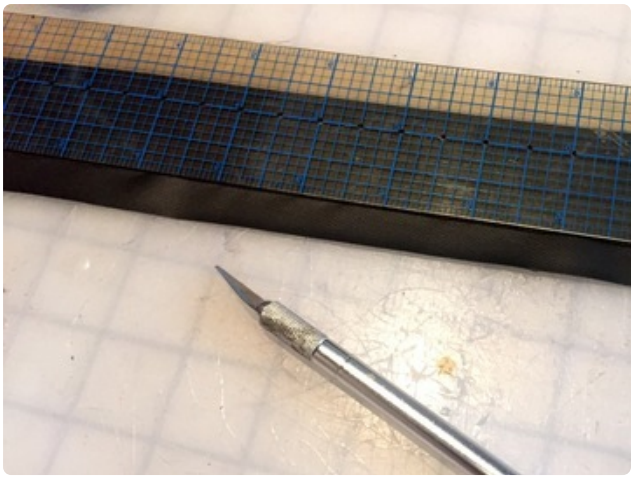
This design looks better when the runes are cut on the back of the acrylic, rather than the front -- it adds to the "floating in space" illusion. To achieve this, remember to **reflect the runes across a vertical axis**, so they're backwards before cutting. Then when you look at them through the acrylic they'll be oriented correctly.

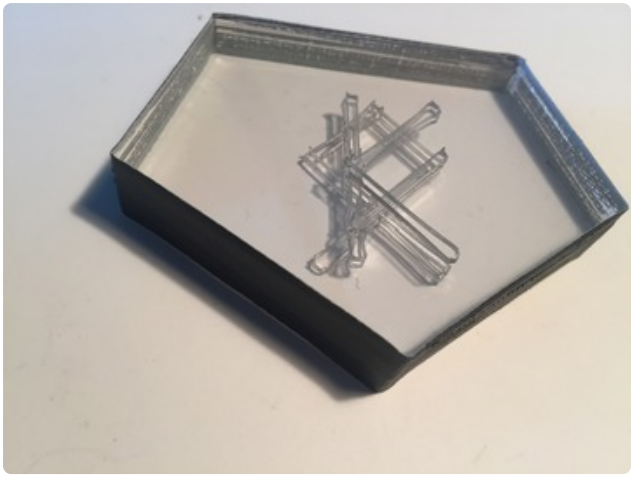


Acrylic Assembly

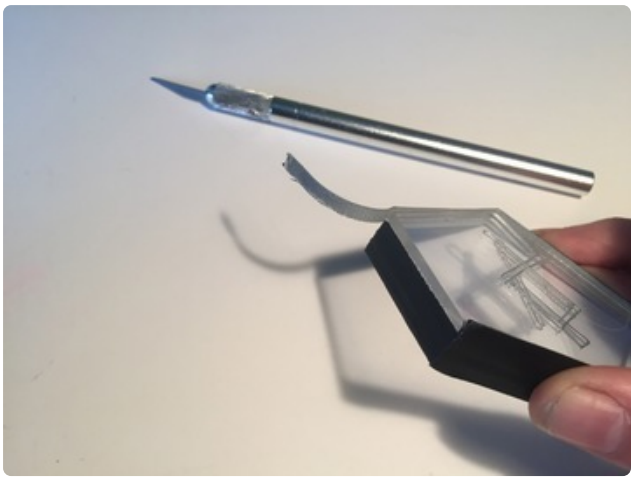
Wear gloves! Fingerprints and cat hair will show up on side-glow acrylic like beacons, so keep it as clean as you can. Mirror Glaze plastic cleaner (https://adafru.it/v6f) can help buff out any fine scratches.

I think it looks best to stack the runes with the most detailed on the bottom and the simplest on the top.

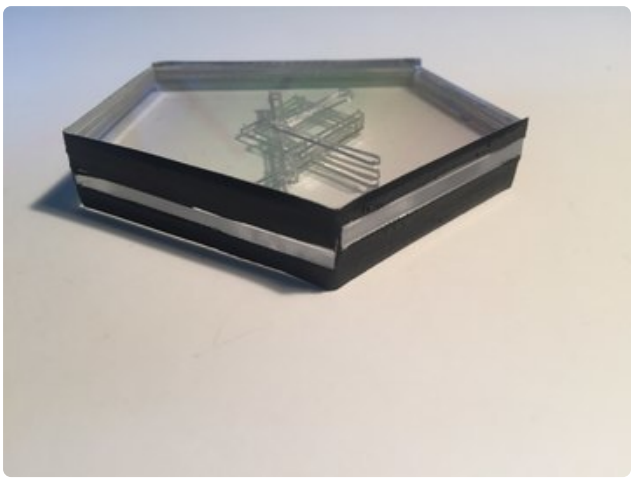


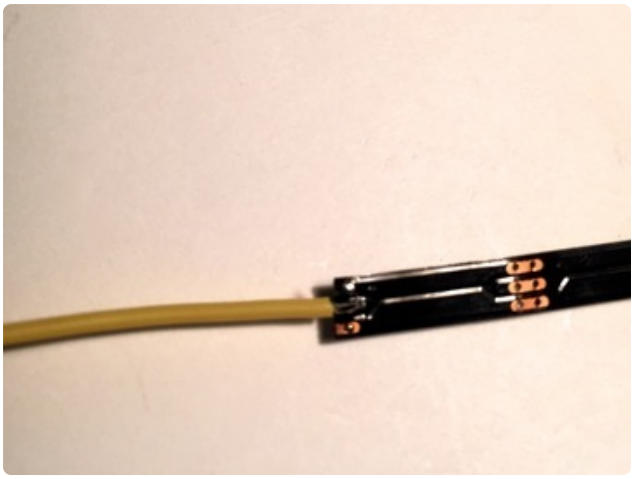


Cut some gaffer's tape to the width of your stack and tape the runes together securely, pressing the tape down hard with a flat metal tool like the back of your scissors.

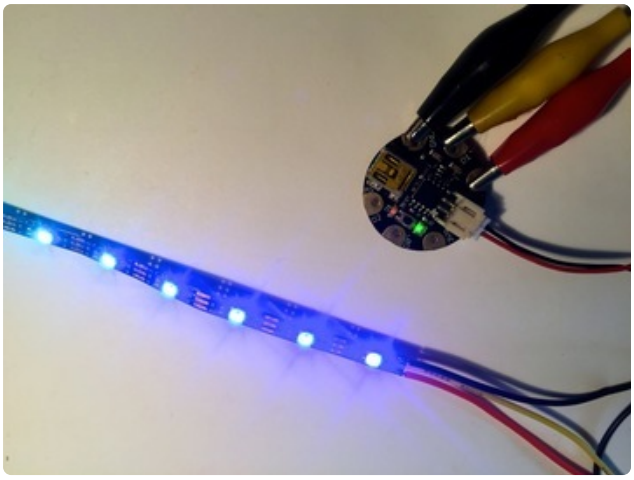


Carefully slice away a thin strip of tape in the places you want the light to shine through. The pixels will wrap in a slight spiral around the stack, so I removed one strip of tape on each side along that spiral.

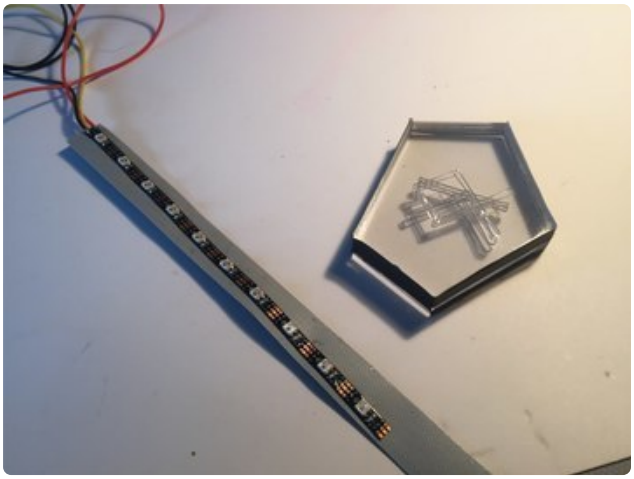




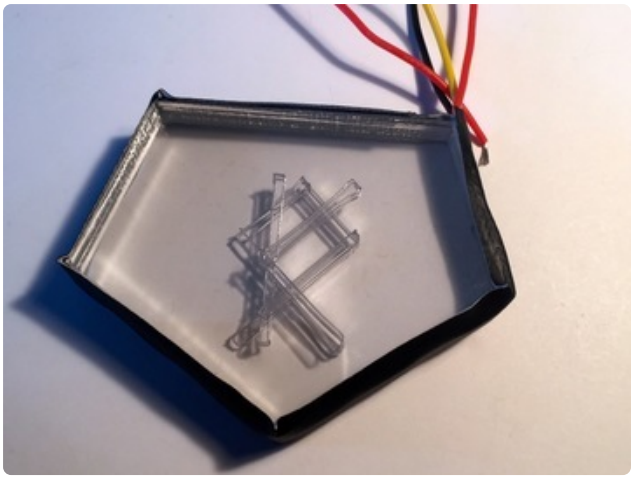
Solder up your neopixels with 3 wires. I found it easiest to solder the data wire (in the middle) on the back of the strip, and the power and ground wires (on the edges) to the front.



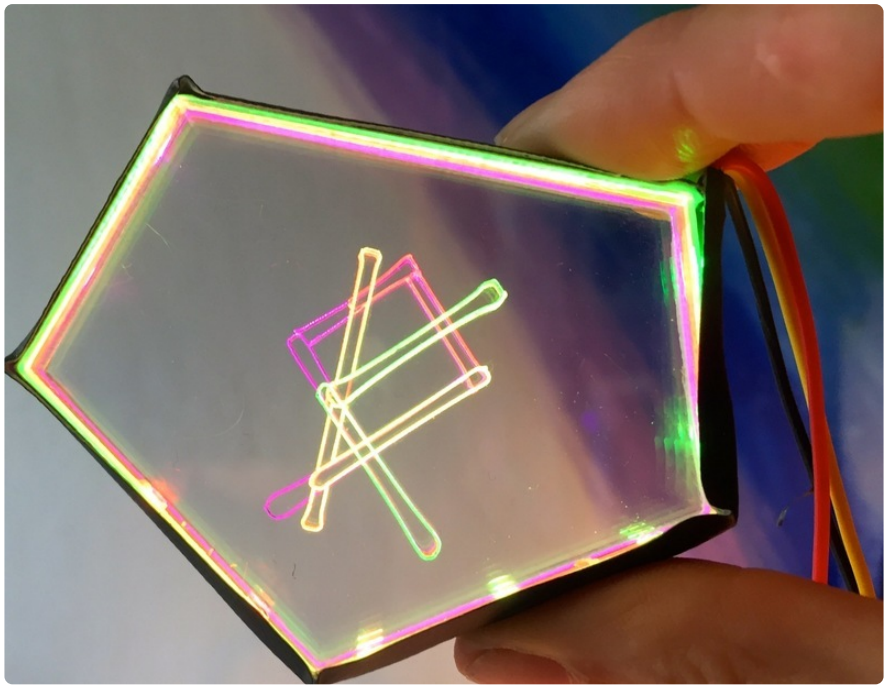
I keep a gemma loaded with neopixel strandtest code handy for testing. Be sure your pixels work before moving on; it's much easier to fix it now.



Cut another piece of Gaffer's tape to slightly wider than the height of your acrylic. Lay the neopixel strand along it diagonally.



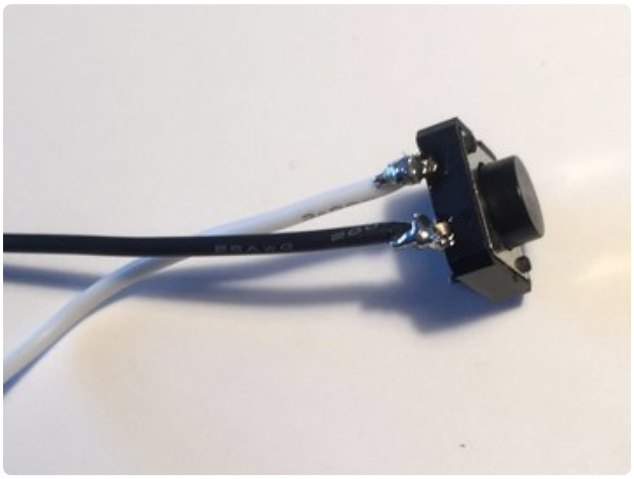
Wrap the tape around the rune stack, lining up all the pixels with the un-covered edges of acrylic.



Hook it back up to your tester and look at the pretty lights! Peek in along the edges to be sure each layer has the neopixels aligned as well as possible.

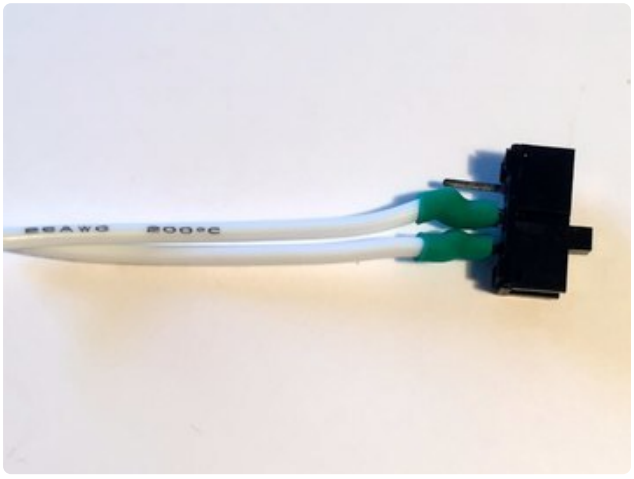
Wiring it Up

Prep



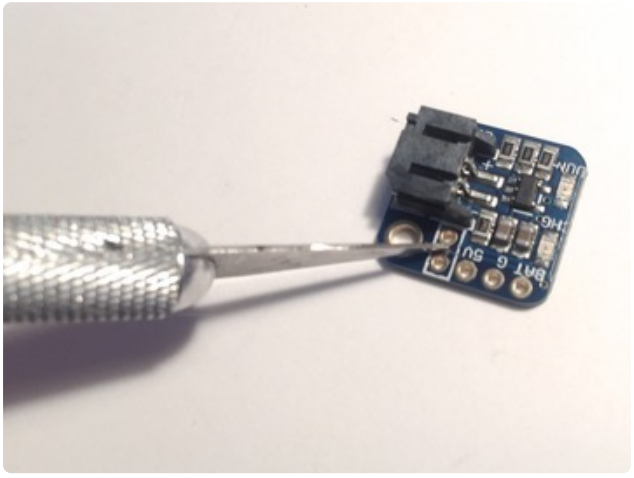
Prep all the elements first:

Add a black and white wire to one side of the momentary switch.

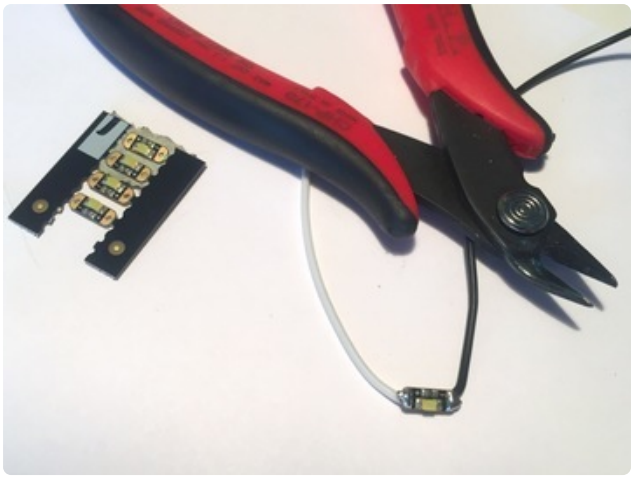


Add two wires to the on/off switch, to the middle leg and one of the side legs. You can cut off the other side leg.

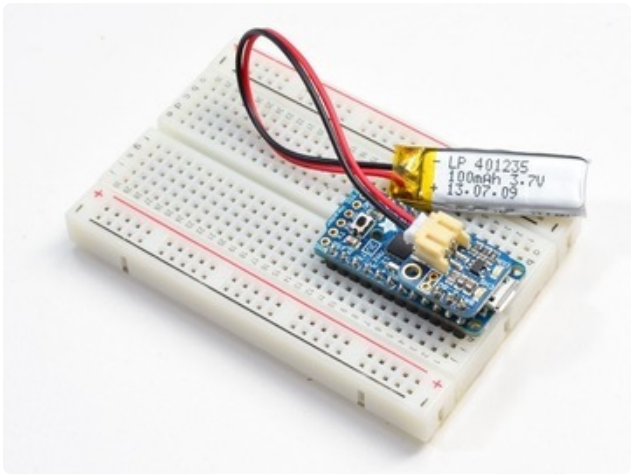
Carefully cut the metal trace between the switch pads on the liPoly backpack charger.



Solder a white wire and a black wire to your LED sequin. This will become the backlight for the button.

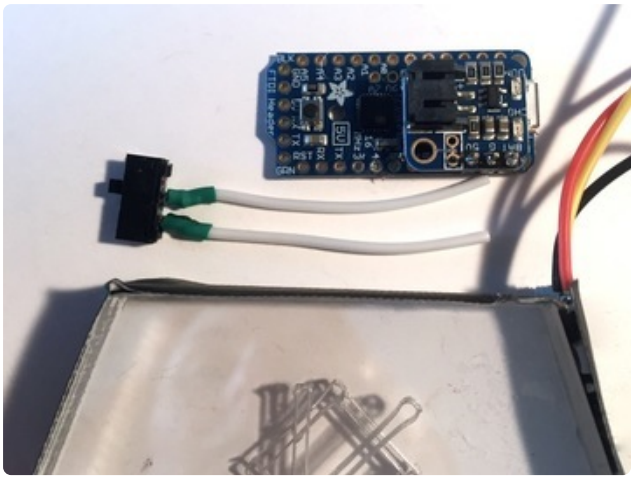


Charger

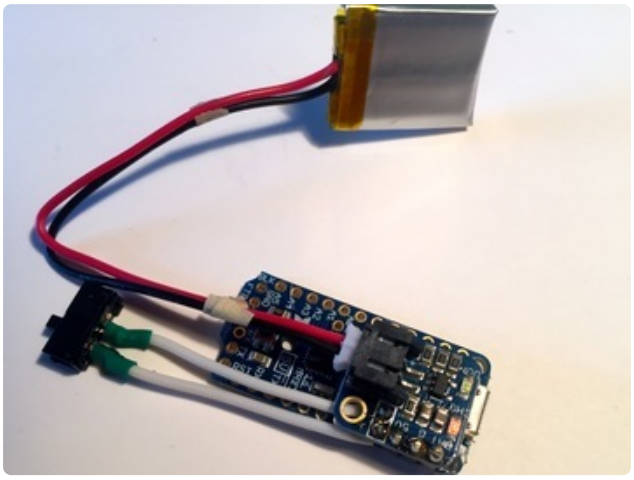


Solder the LiPoly backpack charger to the Pro Trinket using the aligned BAT, G, and 5V pins and the included header.

On/Off Switch

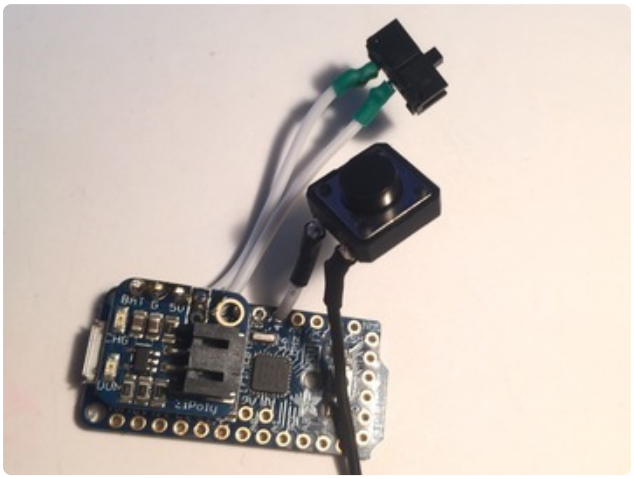


Lay the Pro Trinket along the top of your acrylic and trim the switch's wires to a good length, so it will just reach the edge of your case. Solder the switch wires into the liPoly charger's switch pads.

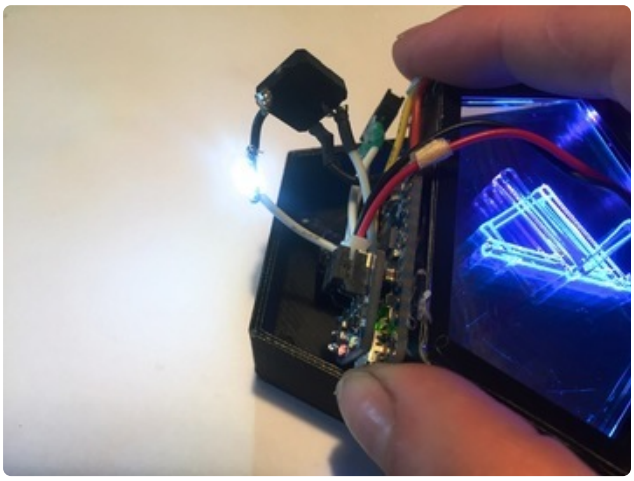


It's easiest to tin the wires and give them a 90 degree bend, then slip them in from underneath. Plug in your battery and flip the switch, making sure the Pro Trinket comes on.

Mode Button

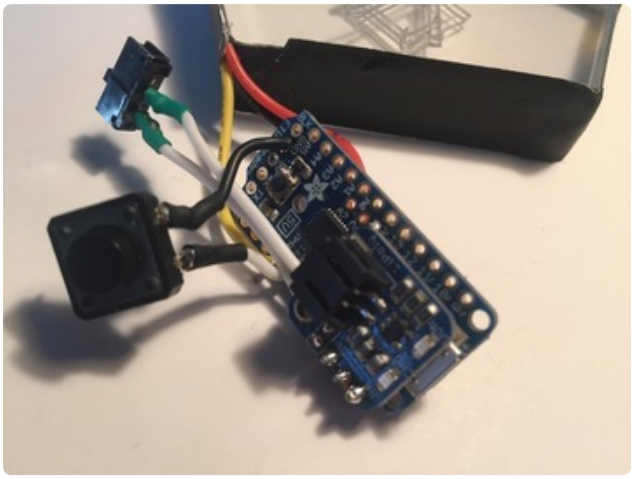


Solder the white wire from the button into pin 4 on the Pro Trinket. Solder the black wire to the G pin near the bottom.

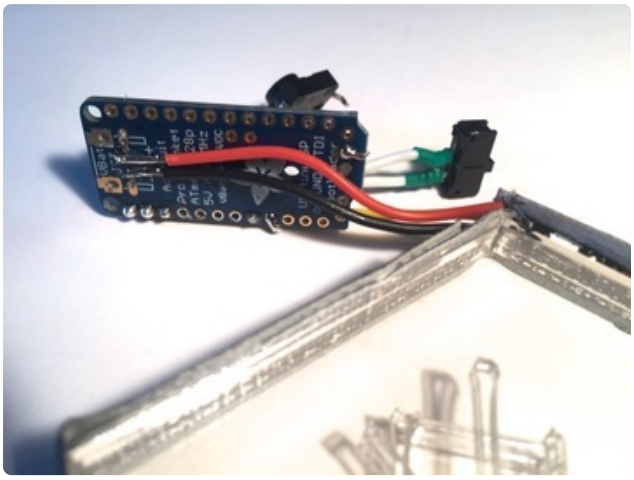


Solder the black wire from your neopixel sequin to the leg on your button opposite the black wire (g). Solder the other wire to pin 5.

Neopixels

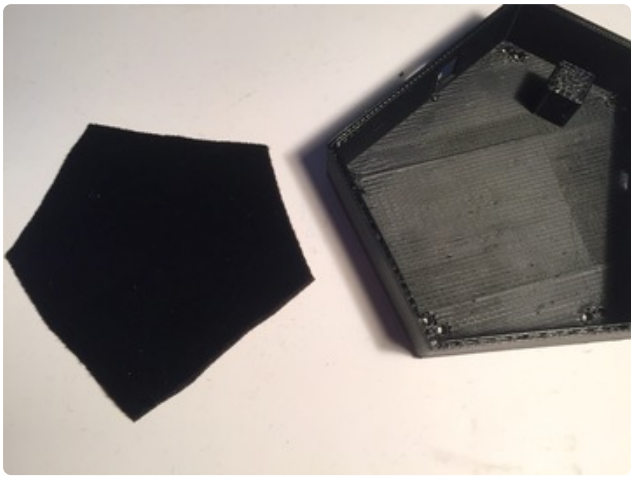


Solder the data line from the neopixels to pin 3.

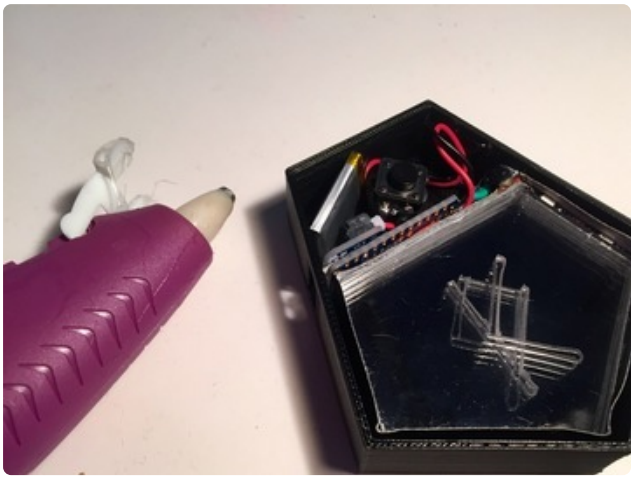


Flip the Pro Trinket over and tin the + and - pads on the back. Solder the + and - wires from the neopixel strand to these pads.

Case Assembly



Use a dab of hot glue to affix the pro trinket to the top side of the acrylic runes so that the USB port is facing outwards.



I added some black velvet to cut down all reflections inside the case. Clean the acrylic once more and place the runes inside, arranging the button and switch and battery to fit neatly. Glue the button and switch in place.

3d Case Design

Here's a video about how I designed the case for this project. You can also download my finished files to print yourself, or order my version of the case from Shapeways.

Edit on Tinkercad

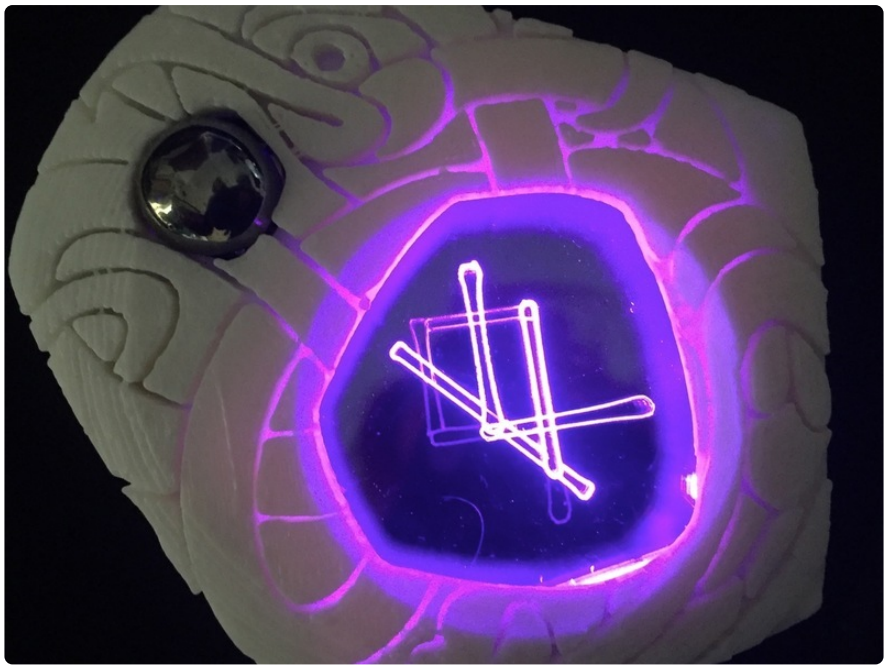
https://adafru.it/19td

Buy from Shapeways

https://adafru.it/vd1

Download from Thingiverse

https://adafru.it/vd2



Finishing the Case

I wanted the lid to look worn and ancient. I tried printing in a few different materials, but ended up going with plain white ABS for a couple reasons: first, the light bleeds through the white ABS in a super cool way.

Second, ABS can be finished with acetone vapors to smooth the edges and make the whole piece look worn down.

I placed my case's lid into an acetone vapor chamber for about 3-4 hours until it became smooth.

Next I sanded it down to get rid of all the shine, and added some nicks and scratches with a file and various other sharp tools I had lying around. I primered it with a paint-on primer from the hardware store -- a spray primer would have gotten into the grooves of the etching and blocked the light from coming through, so I did my best to keep those areas clean.

Primer is key. Don't skip this step! Hardly anything sticks to ABS, and my first attempt had the paint rubbing off after just a day or two of kicking around the house. I want this thing to survive grubby sticky hands at Renaissance faires so a little extra time now is definitely worth while.

I added a base coat of brown acrylic, then a base of Rub n Buff in antique gold (https://adafru.it/vd4). This stuff goes on thick and can be polished and shined

after it dries, so it made a great base coat for underneath the acrylic texture. Wear gloves! This stuff never comes off your hands.

Next, I used a ratty paintbrush and a sponge to dab on low lights in gunmetal gray and metallic black, and then followed that up with some chrome silver highlights. Once I was happy with the look, I sealed the whole thing sith some spray polyurethane sealant since I expect this thing to go through some wear and tear.

Find a glass gem that fits inside the button hole without falling out. Craft stores have bags and bags of these in the floral section -- they're often used for filling vases to support flowers. Each one is a slightly different size and color, so it may take a few tries to find the right one.

My LED sequin needed a little more diffusing, so I soaked a couple circles of paper in crazy glue and affixed them to the bottom of the gem. To keep the light from bleeding through the case, I cut a piece of black vinyl shelf liner to fit around the gem and cover the electronics inside.