LED Stego Flex Spike Hoodie

Created by Becky Stern

https://learn.adafruit.com/led-stego-flex-spike-hoodie

Last updated on 2023-08-29 02:31:25 PM EDT

Table of Contents

Overview

Dress up as a time-traveling dinosaur with these glowing stego spikes! This easy project mashes up 3D printing and sewing to make your own super-custom flexible spiky hooded sweatshirt.

For this project you will need:

- 12mm diffused flat digital pixel strand ()
- FLORA main board ()
- White NinjaFlex 3D printing filament ()
- USB battery pack () and cable
- Hooded sweatshirt
- Plain cotton or cotton/poly sewing thread

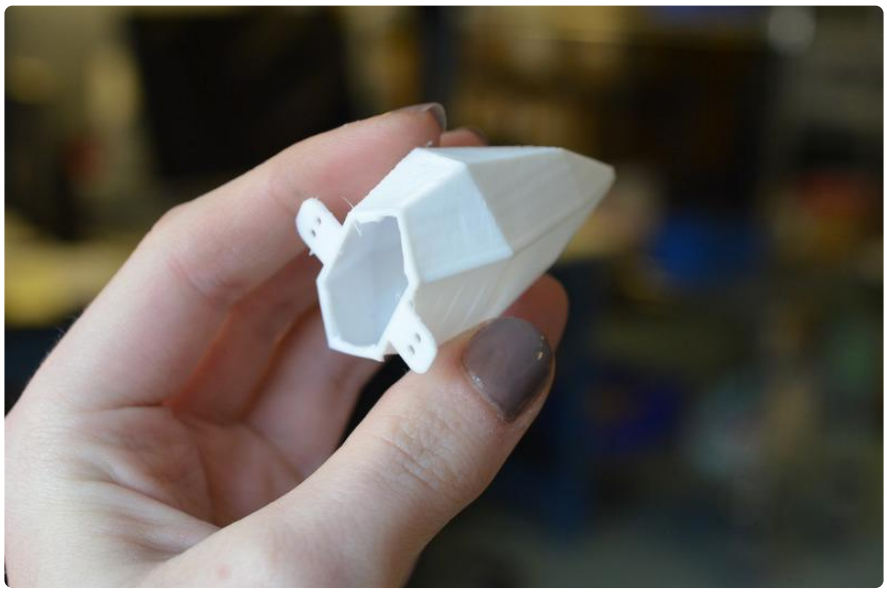
tools:

- Soldering iron ()
- Wire strippers ()
- Flush diagonal cutters ()
- Scissors
- Sewing needle ()

Like this project?

Check out Matt Pinner's Stego Hoodie - which inspired this build! ()

3D Print Spikes



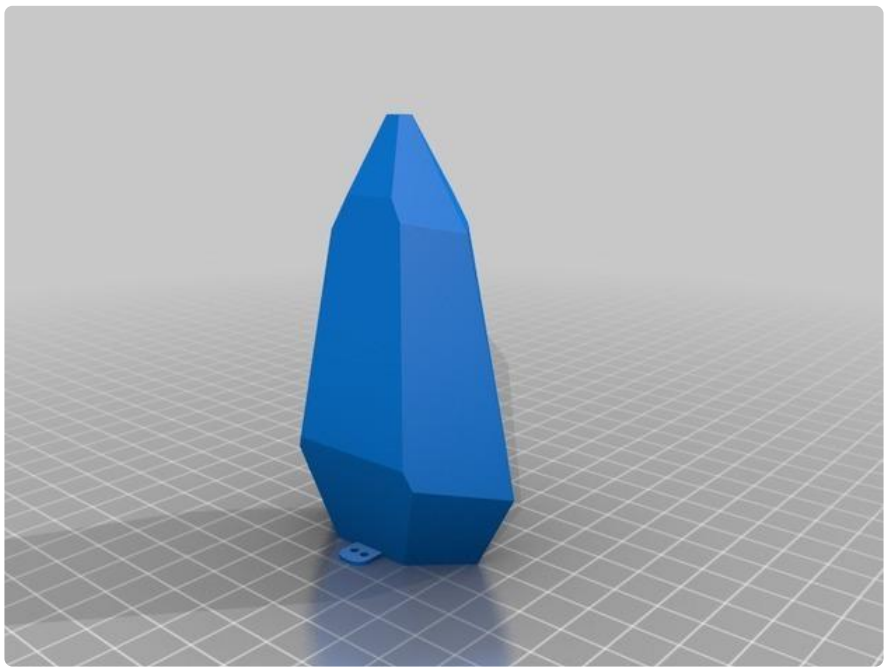
Parts are optimized to print on a Makerbot Replicator 2 () and sliced with Makerware. Download the parts from thingiverse ()and print them out using the recommended settings below.

Download STLs

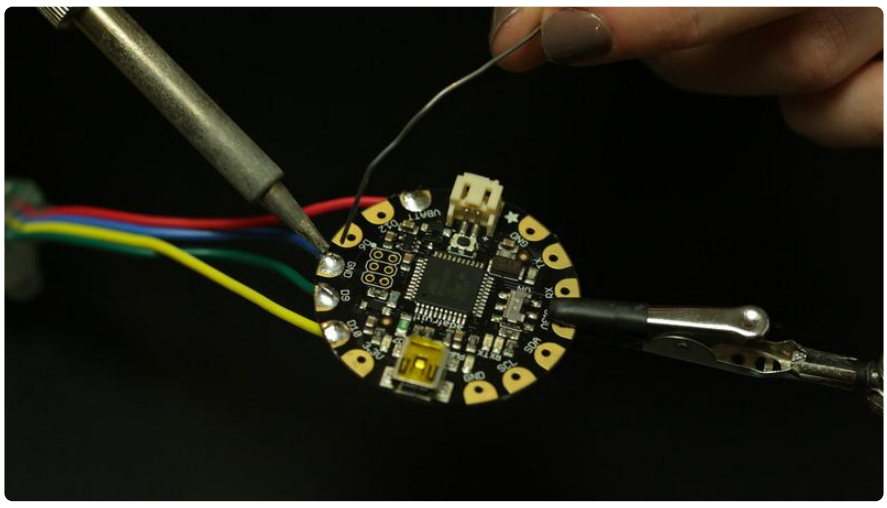
NinjaFlex

The spikes are hollow and designed to be printed in Ninjaflex filament (). The STL files come in three different size, large, medium and small. Each spike includes two sewable tabs for easily attaching to your hoodie or costume.

stegospike-large.stl stegospike-medium.stl stegospike-small.stl	Ninjaflex @225 %15 infill 2 shells 0.2 layer height 45/150 speeds	Takes about 10-15 minutes for each spike



Assemble Circuit



Read the 12mm pixels guide () to find the input end of your pixel strand. Clip off the connector and solder the four wires to FLORA as follows:
red wire -> VBATT
blue wire -> GND
green wire -> D9
yellow wire -> D10

Don't forget to download the WS2801 library from the 12mm pixels guide (). Below is a sample sketch for your dino spikes incuding random flashing and changing colors--copy it into your Adafruit Arduino IDE as-is and then mod the colors and animation to make it your own. Remember that to program FLORA you need to download the special Adafruit version of the Arduino IDE from the Getting Started with FLORA guide ().

```
#include "SPI.h"
#include "Adafruit_WS2801.h"

/*****************************************************************************
Example sketch for driving Adafruit WS2801 pixels!


  Designed specifically to work with the Adafruit RGB Pixels!
  12mm Bullet shape ----&gt; https://www.adafruit.com/products/322
  12mm Flat shape   ----&gt; https://www.adafruit.com/products/738
  36mm Square shape ----&gt; https://www.adafruit.com/products/683

  These pixels use SPI to transmit the color data, and have built in
  high speed PWM drivers for 24 bit color per pixel
  2 pins are required to interface

  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  BSD license, all text above must be included in any redistribution

*****************************************************************************/

// Choose which 2 pins you will use for output.
// Can be any valid output pins.
// The colors of the wires may be totally different so
// BE SURE TO CHECK YOUR PIXELS TO SEE WHICH WIRES TO USE!
uint8_t dataPin  = 10;    // Yellow wire on Adafruit Pixels
uint8_t clockPin = 9;    // Green wire on Adafruit Pixels

Adafruit_WS2801 strip = Adafruit_WS2801(19, dataPin, clockPin);

// Here is where you can put in your favorite colors that will appear!
// just add new {nnn, nnn, nnn}, lines. They will be picked out randomly
//                            R   G   B
uint8_t myColors[][3] = {{232, 100, 255},   // purple
                         {200, 200, 20},   // yellow
                         {30, 200, 200},   // blue
                          };

// don't edit the line below
#define FAVCOLORS sizeof(myColors) / 3

void setup() {
```

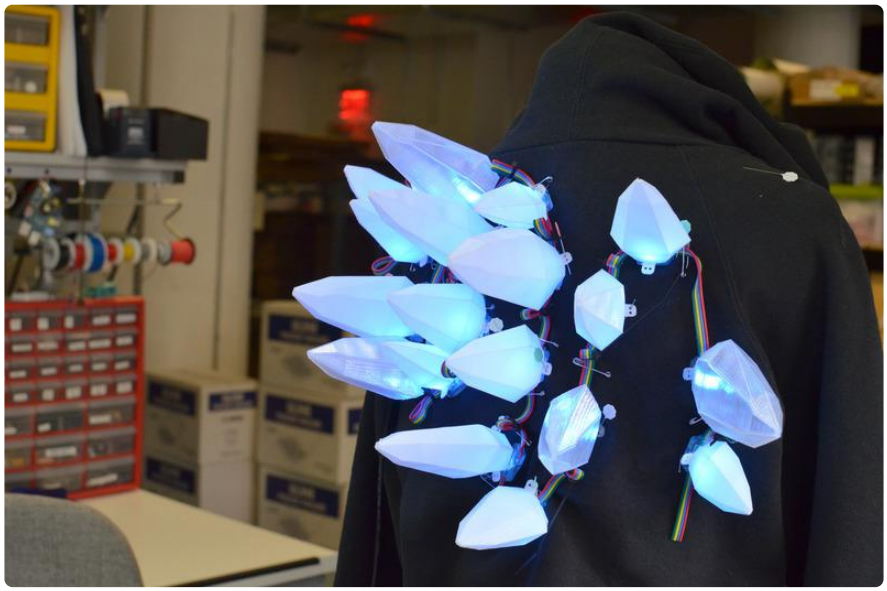
```
  strip.begin();
  strip.show(); // Initialize all pixels to 'off'
}

void loop() {
flashRandom(5, 8);  // first number is 'wait' delay, shorter num == shorter twinkle
flashRandom(5, 5);  // second number is how many neopixels to simultaneously light
up
flashRandom(5, 11);
flashRandom(5, 10);
flashRandom(5, 9);
flashRandom(5, 7);
colorWipe(Color(232, 100, 255), 50); // Red
colorWipe(Color(200, 200, 20), 50); // Green
colorWipe(Color(30, 200, 200), 50); // Blue
rainbowCycle(10);
colorWipe(Color(0, 0, 0), 50); // Red
}

// Fill the dots one after the other with a color
void colorWipe(uint32_t c, uint8_t wait) {
  for(uint16_t i=0; i&lt;strip.numPixels(); i++) {
      strip.setPixelColor(i, c);
      strip.show();
      delay(wait);
  }
}

void rainbow(uint8_t wait) {
  uint16_t i, j;

  for(j=0; j&lt;256; j++) {
    for(i=0; i&lt;strip.numPixels(); i++) {
      strip.setPixelColor(i, Wheel((i+j) &amp; 255));
    }
    strip.show();
    delay(wait);
  }
}

// Slightly different, this makes the rainbow equally distributed throughout
void rainbowCycle(uint8_t wait) {
  uint16_t i, j;

  for(j=0; j&lt;256*5; j++) { // 5 cycles of all colors on wheel
    for(i=0; i&lt; strip.numPixels(); i++) {
      strip.setPixelColor(i, Wheel(((i * 256 / strip.numPixels()) + j) &amp; 255));
    }
    strip.show();
    delay(wait);
  }
}

// Input a value 0 to 255 to get a color value.
// The colours are a transition r - g - b - back to r.
uint32_t Wheel(byte WheelPos) {
  if(WheelPos &lt; 85) {
   return Color(WheelPos * 3, 255 - WheelPos * 3, 0);
  } else if(WheelPos &lt; 170) {
   WheelPos -= 85;
   return Color(255 - WheelPos * 3, 0, WheelPos * 3);
  } else {
   WheelPos -= 170;
   return Color(0, WheelPos * 3, 255 - WheelPos * 3);
  }
}

void flashRandom(int wait, uint8_t howmany) {
```

```
  for(uint16_t i=0; i&lt;howmany; i++) {
    // pick a random favorite color!
    int c = random(FAVCOLORS);
    int red = myColors[c][0];
    int green = myColors[c][1];
    int blue = myColors[c][2];

    // get a random pixel from the list
    int j = random(strip.numPixels());

    // now we will 'fade' it in 5 steps
    for (int x=0; x &lt; 5; x++) {
      int r = red * (x+1); r /= 5;
      int g = green * (x+1); g /= 5;
      int b = blue * (x+1); b /= 5;

      strip.setPixelColor(j, Color(r, g, b));
      strip.show();
      delay(wait);
    }
    // &amp; fade out in 5 steps
    for (int x=5; x &gt;= 0; x--) {
      int r = red * x; r /= 5;
      int g = green * x; g /= 5;
      int b = blue * x; b /= 5;

      strip.setPixelColor(j, Color(r, g, b));
      strip.show();
      delay(wait);
    }
  }
  // LEDs will be off when done (they are faded to 0)
}

/* Helper functions */

// Create a 24 bit color value from R,G,B
uint32_t Color(byte r, byte g, byte b)
{
  uint32_t c;
  c = r;
  c &lt;&lt;= 8;
  c |= g;
  c &lt;&lt;= 8;
  c |= b;
  return c;
}
```

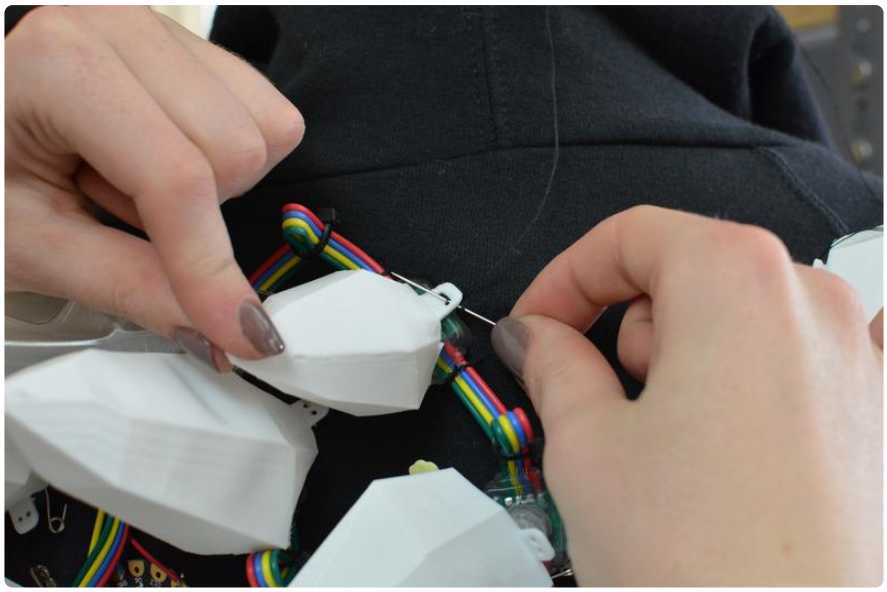
Layout Design and Sew

Mock up your layout using safety pins to attach the pixels and spikes to your hoodie.

You can use zipties to cinch up the wires between pixels to put them closer together. There is no right way to arrange your spikes! Put a line of them up the back of the hood, cluster them on one shoulder, or any other arrangement you can think up. You can vary the size of the spikes and even try printing in a different color material (in the photo above, a few spikes are printed in clear NinjaFlex).



When you're happy with the design, clip off the unused pixels with flush diagonal cutters, and stitch the pixels and spikes to the hoodie one by one.



Take a reference photo if you want to remove all the spikes, or just stitch one on at a time so you don't forget where each one goes.

Use a USB battery pack to power your hoodie-- you can stash it in the pocket and run a USB cable to FLORA. Keep in mind that FLORA's onboard switch does not control USB power, and so to turn off you may have to disconnect the USB cable.