Glowing LED Chair

Created by Ruiz Brothers

https://learn.adafruit.com/glowing-led-chair

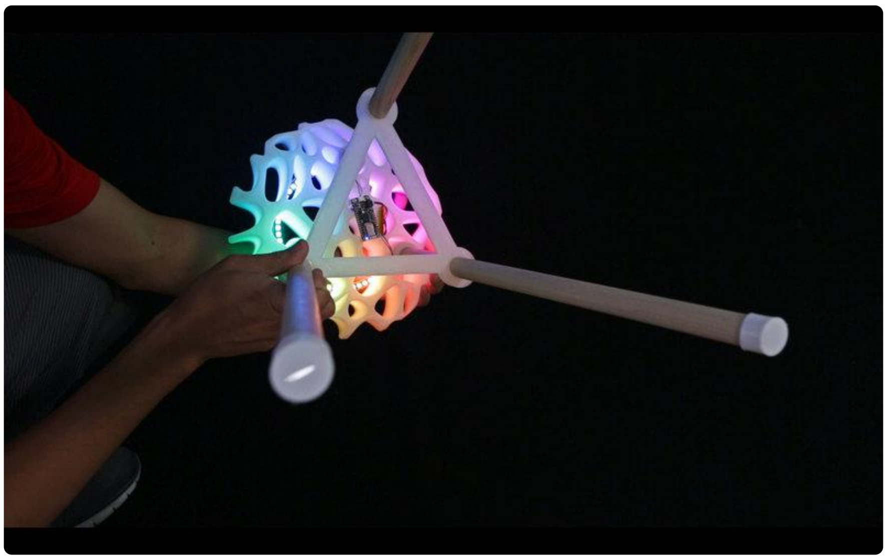
Last updated on 2023-08-29 03:14:42 PM EDT

Table of Contents

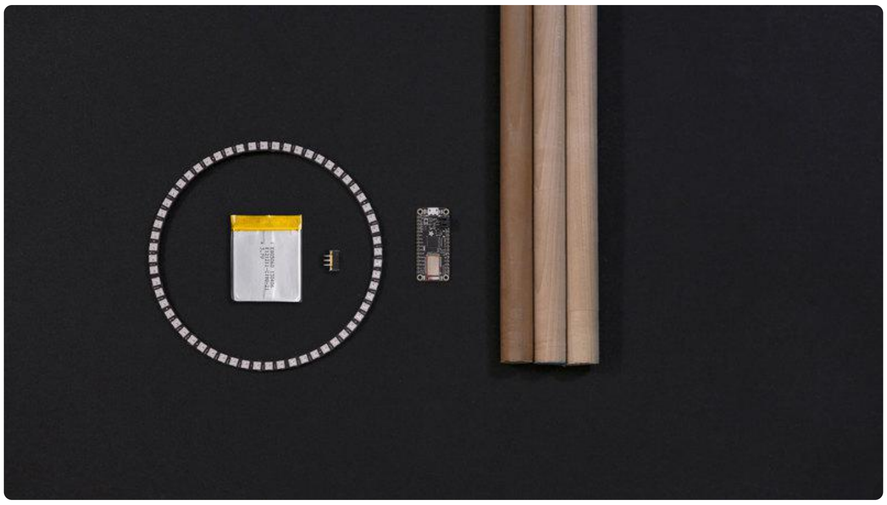
Overview

Light up the night and relax on a comfortable seat you can easy take to your next night out by the fire.

These super bright NeoPixel LEDs let you change colors and animations with the Adafruit Bluefruit Connect LE app. This customizable 3D printed design can easily hold up to 200 lbs of weight.



All of the electronics are mounted inside and on the bottom of the seat with plenty of room to add custom parts to the design.



Prerequisite Guides

We suggest walking through the following tutorials to get a better understanding of NeoPixel LEDs and soldering.

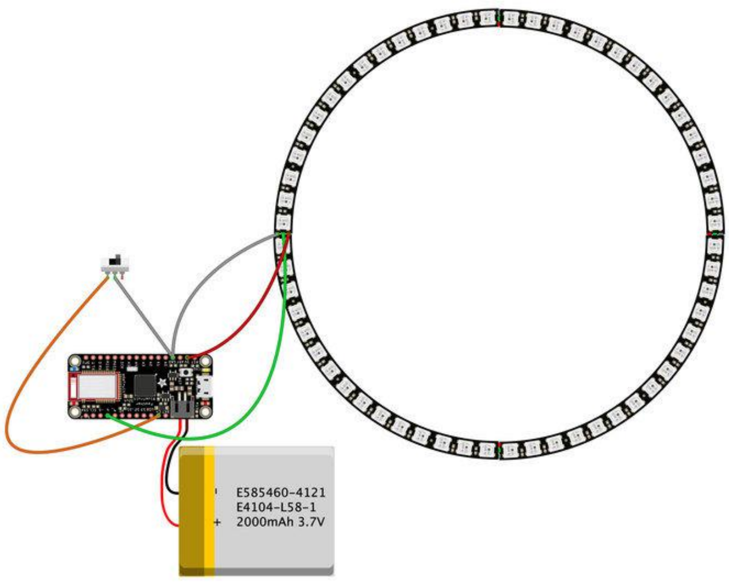
- Collin's Lab: Soldering ()
- Adafruit NeoPixel Uber Guide ()
- Adafruit PowerBoost 1000C ()

Parts, Tools and Supplies

You'll need the following tools, parts and supplies to complete this build.

- 4x 1/4 60 Ring (4 make a full circle) (http://adafru.it/1768)
- 2000mAh Lithium Ion Polymer Battery (http://adafru.it/2011)
- Adafruit Feather 32u4 Bluefruit LE (http://adafru.it/2829)
- 7/8 inch Dowel
- Silicone wires ()
- soldering iron and solder ()
- wire strippers ()
- diagonal flush snips ()
- helping third hand tool ()
- Panavise ()
- 4x M2.5x.5x5mm phillips flat head screws ()

Circuit Diagram



Wired Connections

The circuit diagram above shows how the components will be wired together. This won't be 100% exact in the actual circuit but it's a very close approximation.

Solder all of the sides together except the last DIN and DOUT, leave those two last ones separated. DIN is where the data will feed into the neopixel ring.

- 5V+ Power on NeoPixel ring to BAT on Adafruit Feather.
- Ground on NeoPixel ring to GND on Adafruit Feather.
- DIN on NeoPixel ring to Pin 6 on Adafruit Feather.

Code

Arduino Libraries

To use the Daftpunk BLE sketch you'll want to make sure you're using the latest version of the Arduino IDE () (1.6.12 at the time of this writing).

If you're totally new to Arduino take a little time to go through some introductory tutorials like how to make a LED blink (). This will help you understand how to use the IDE, load a sketch, and upload code.

Next you'll need to make sure the libraries used by the sketch are installed. With the latest Arduino IDE you can use its library manager () to easily install libraries, or check out this guide on how to manually install a library (). You'll want to install the following libraries:

- Adafruit BluefruitLE nRF51
- Adafruit NeoPixel

Search for the libraries in the library manager and they should be easy to find and install.

Adafruit AVR Boards

Next, you'll need to install the Adafruit AVR boards package from the Boards Manager. Open the Boards Manager and search for Adafruit AVR. This includes all of the boards from Adafruit and will make Arduino compatabile with them. The Daftpunk BLE sketch was tested with version 1.4.0.

Uploading Sketch to Adafruit Feather BLE

This sketch will run the Bluetooth controlled LED program to the NeoPixel strips that are mounted to the front of the visor.

To load the sketch make sure the libraries above are installed, and the Arduino is connected to the computer through a USB cable. Under the Tools -> Board menu make sure the Adafruit Feather 32u4 is selected, and under the Tools -> Port menu the serial port for the Adafruit Feather is selected.

Then press the upload button or click the Sketch -> Upload item to send the code to the Arduino. Woo-hoo the sketch should be running.

Connect Adafruit BLE Mobile App to Adafruit Feather BLE

Download the Adafruit BLE Connect app for iOS or Android. Under the peripherals list, tap theconnect button on the Adafruit Bluefruit LE item. Make sure the Feather board is powered on. Select "Controller" and choose either Control Pad or the Color Picker.

- Adafruit Bluefruit LE Connect for iOS ()
- Adafruit Bluefruit LE Connect for Android ()

Control Pad

Buttons 1-4 will trigger an animation.

1. larsonScanner
2. color wipe
3. rainbow gradient
4. rainbow cycle

Color Picker

Here you can change the brightness or RGB value of the leds.

feather_bluefruit_neopixel_animation_

Make sure to have a data cable and not a power only USB cable.

You'll want to update the number of neopixels to 60.

Change this line to:

#define NUMPIXELS   60

Check the sketch below

```
    /*********************************************************************
 This is an example for our nRF51822 based Bluefruit LE modules

 Pick one up today in the adafruit shop!
 Adafruit invests time and resources providing this open source code,
 please support Adafruit and open-source hardware by purchasing
 products from Adafruit!

 MIT license, check LICENSE for more information
 All text above, and the splash screen below must be included in
 any redistribution
*********************************************************************/

#include &lt;string.h&gt;
#include &lt;Arduino.h&gt;
#include &lt;SPI.h&gt;
#if not defined (_VARIANT_ARDUINO_DUE_X_) &amp;&amp; not defined
(_VARIANT_ARDUINO_ZERO_)
  #include &lt;SoftwareSerial.h&gt;
#endif

#include "Adafruit_BLE.h"
#include "Adafruit_BluefruitLE_SPI.h"
#include "Adafruit_BluefruitLE_UART.h"

#include "BluefruitConfig.h"

#include &lt;Adafruit_NeoPixel.h&gt;

/*=========================================================================
    APPLICATION SETTINGS

    FACTORYRESET_ENABLE       Perform a factory reset when running this sketch

                              Enabling this will put your Bluefruit LE module
                              in a 'known good' state and clear any config
                              data set in previous sketches or projects, so
                              running this at least once is a good idea.

                              When deploying your project, however, you will
                              want to disable factory reset by setting this
                              value to 0.  If you are making changes to your
                              Bluefruit LE device via AT commands, and those
                              changes aren't persisting across resets, this
                              is the reason why.  Factory reset will erase
                              the non-volatile memory where config data is
                              stored, setting it back to factory default
                              values.

                              Some sketches that require you to bond to a
                              central device (HID mouse, keyboard, etc.)
                              won't work at all with this feature enabled
                              since the factory reset will clear all of the
                              bonding data stored on the chip, meaning the
                              central device won't be able to reconnect.
    PIN                       Which pin on the Arduino is connected to the
NeoPixels?
    NUMPIXELS                 How many NeoPixels are attached to the Arduino?
    -----------------------------------------------------------------------*/
    #define FACTORYRESET_ENABLE     1

    #define PIN                     6
    #define NUMPIXELS               60
/*=========================================================================*/

Adafruit_NeoPixel pixel = Adafruit_NeoPixel(NUMPIXELS, 6); // NeoPixel Object for
Visor Strips

// Create the bluefruit object, either software serial...uncomment these lines
/*
SoftwareSerial bluefruitSS = SoftwareSerial(BLUEFRUIT_SWUART_TXD_PIN,
BLUEFRUIT_SWUART_RXD_PIN);
```

```
Adafruit_BluefruitLE_UART ble(bluefruitSS, BLUEFRUIT_UART_MODE_PIN,
                      BLUEFRUIT_UART_CTS_PIN, BLUEFRUIT_UART_RTS_PIN);
*/

/* ...or hardware serial, which does not need the RTS/CTS pins. Uncomment this line
*/
// Adafruit_BluefruitLE_UART ble(BLUEFRUIT_HWSERIAL_NAME, BLUEFRUIT_UART_MODE_PIN);

/* ...hardware SPI, using SCK/MOSI/MISO hardware SPI pins and then user selected CS/
IRQ/RST */
Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_CS, BLUEFRUIT_SPI_IRQ,
BLUEFRUIT_SPI_RST);

/* ...software SPI, using SCK/MOSI/MISO user-defined SPI pins and then user
selected CS/IRQ/RST */
//Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_SCK, BLUEFRUIT_SPI_MISO,
//                             BLUEFRUIT_SPI_MOSI, BLUEFRUIT_SPI_CS,
//                             BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);


// A small helper
void error(const __FlashStringHelper*err) {
  Serial.println(err);
  while (1);
}

// function prototypes over in packetparser.cpp
uint8_t readPacket(Adafruit_BLE *ble, uint16_t timeout);
float parsefloat(uint8_t *buffer);
void printHex(const uint8_t * data, const uint32_t numBytes);

// the packet buffer
extern uint8_t packetbuffer[];


/**************************************************************************/
/*!
    @brief  Sets up the HW an the BLE module (this function is called
            automatically on startup)
*/
/**************************************************************************/
//additional variables

//Color
    uint8_t red = 255;
    uint8_t green = 255;
    uint8_t blue = 255;
    uint8_t animationState = 1;

    int pos = 0, dir = 1; // Position, direction of "eye" for larson scanner
animation

void setup(void)
{
  //while (!Serial);  // required for Flora &amp; Micro
  delay(500);

  // turn off neopixel
  pixel.begin(); // This initializes the NeoPixel library.

  for(uint8_t i=0; i&lt;NUMPIXELS; i++) {
    pixel.setPixelColor(i, pixel.Color(0,0,0)); // off
  }
  colorWipe(pixel.Color(255, 255, 255), 20);
  colorWipe(pixel.Color(0, 0, 0), 20);
  pixel.show();

  Serial.begin(115200);
  Serial.println(F("Adafruit Bluefruit Neopixel Color Picker Example"));
  Serial.println(F("-----------------------------------------------"));

  /* Initialise the module */
  Serial.print(F("Initialising the Bluefruit LE module: "));

  if ( !ble.begin(VERBOSE_MODE) )
  {
    error(F("Couldn't find Bluefruit, make sure it's in CoMmanD mode &amp; check 
wiring?"));
  }
  Serial.println( F("OK!") );

  if ( FACTORYRESET_ENABLE )
  {
    /* Perform a factory reset to make sure everything is in a known state */
    Serial.println(F("Performing a factory reset: "));
    if ( ! ble.factoryReset() ){
      error(F("Couldn't factory reset"));
    }
  }

  /* Disable command echo from Bluefruit */
  ble.echo(false);

  Serial.println("Requesting Bluefruit info:");
  /* Print Bluefruit information */
  ble.info();

  Serial.println(F("Please use Adafruit Bluefruit LE app to connect in Controller 
mode"));
  Serial.println(F("Then activate/use the sensors, color picker, game controller, 
etc!"));
  Serial.println();

  ble.verbose(false);  // debug info is a little annoying after this point!

  /* Wait for connection */
  while (! ble.isConnected()) {
      delay(500);
  }

  Serial.println(F("***********************"));

  // Set Bluefruit to DATA mode
  Serial.println( F("Switching to DATA mode!") );
  ble.setMode(BLUEFRUIT_MODE_DATA);

  Serial.println(F("***********************"));
}

/**************************************************************************/
/*!
    @brief  Constantly poll for new command or response data
*/
/**************************************************************************/
void loop(void) {
  /* Wait for new data to arrive */
  uint8_t len = readPacket(&amp;ble, BLE_READPACKET_TIMEOUT);
  if (len == 0) return;

  /* Got a packet! */
  // printHex(packetbuffer, len);

  // Color
  if (packetbuffer[1] == 'C') {
    uint8_t red = packetbuffer[2];
    uint8_t green = packetbuffer[3];
    uint8_t blue = packetbuffer[4];
    Serial.print ("RGB #");
    if (red &lt; 0x10) Serial.print("0");
    Serial.print(red, HEX);
    if (green &lt; 0x10) Serial.print("0");
    Serial.print(green, HEX);
    if (blue &lt; 0x10) Serial.print("0");
    Serial.println(blue, HEX);

    for(uint8_t i=0; i&lt;NUMPIXELS; i++) {
      pixel.setPixelColor(i, pixel.Color(red,green,blue));
    }
    pixel.show(); // This sends the updated pixel color to the hardware.
  }

  // Buttons
  if (packetbuffer[1] == 'B') {

    uint8_t buttnum = packetbuffer[2] - '0';
    boolean pressed = packetbuffer[3] - '0';
    Serial.print ("Button "); Serial.print(buttnum);
    animationState = buttnum;
    if (pressed) {
      Serial.println(" pressed");
    } else {
      Serial.println(" released");
    }

  if (animationState == 1){
    for(uint16_t i=0; i&lt;pixel.numPixels(); i++) {
      pixel.setPixelColor(i, pixel.Color(0,0,0));
  }
    larsonScanner(pixel.Color(255,0,0), 30);
    pixel.show(); // This sends the updated pixel color to the hardware.
  }

  if (animationState == 2){
    colorWipe(pixel.Color(100, 0, 0), 20);
    colorWipe(pixel.Color(0, 0, 0), 20);
    colorWipe(pixel.Color(100, 0, 0), 20);
    colorWipe(pixel.Color(0, 0, 0), 20);
    colorWipe(pixel.Color(100, 0, 0), 20);
    colorWipe(pixel.Color(0, 0, 0), 20);
    pixel.show(); // This sends the updated pixel color to the hardware.
  }

  if (animationState == 3){
    for(uint16_t i=0; i&lt;pixel.numPixels(); i++) {
      pixel.setPixelColor(i, pixel.Color(0,0,0));
    }
    pixel.setBrightness(255);
    rainbow(10);
    pixel.show(); // This sends the updated pixel color to the hardware.
  }

  if (animationState == 4){
    for(uint16_t i=0; i&lt;pixel.numPixels(); i++) {
      pixel.setPixelColor(i, pixel.Color(0,0,0));
    }
    pixel.setBrightness(255);
    rainbowCycle(10);
    pixel.show(); // This sends the updated pixel color to the hardware.
  }
 }
}


// 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;pixel.numPixels(); i++) {
      pixel.setPixelColor(i, c);
      pixel.show();
      delay(wait);
  }
}

void larsonScanner(uint32_t c, uint8_t wait){
   int j;

 for(uint16_t i=0; i&lt;pixel.numPixels()+5; i++) {
  // Draw 5 pixels centered on pos.  setPixelColor() will clip any
  // pixels off the ends of the strip, we don't need to watch for that.
  pixel.setPixelColor(pos - 2, 0x100000); // Dark red
  pixel.setPixelColor(pos - 1, 0x800000); // Medium red
  pixel.setPixelColor(pos , 0xFF3000); // Center pixel is brightest
  pixel.setPixelColor(pos + 1, 0x800000); // Medium red
  pixel.setPixelColor(pos + 2, 0x100000); // Dark red

  pixel.show();
  delay(wait);

  // Rather than being sneaky and erasing just the tail pixel,
  // it's easier to erase it all and draw a new one next time.
  for(j=-2; j&lt;= 2; j++) pixel.setPixelColor(pos+j, 0);

  // Bounce off ends of strip
  pos += dir;
  if(pos &lt; 0) {
    pos = 1;
    dir = -dir;
  } else if(pos &gt;= pixel.numPixels()) {
    pos = pixel.numPixels() - 2;
    dir = -dir;
  }
 }
 //colorWipe(pixel.Color(0, 0, 0), 20);
}



void flashRandom(int wait, uint8_t howmany) {

  for(uint16_t i=0; i&lt;howmany; i++) {
    // get a random pixel from the list
    int j = random(pixel.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;

      pixel.setPixelColor(j, pixel.Color(r, g, b));
      pixel.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;

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

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

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

//Theatre-style crawling lights.
void theaterChase(uint32_t c, uint8_t wait) {
  for (int j=0; j&lt;10; j++) {  //do 10 cycles of chasing
    for (int q=0; q &lt; 3; q++) {
      for (int i=0; i &lt; pixel.numPixels(); i=i+3) {
        pixel.setPixelColor(i+q, c);    //turn every third pixel on
      }
      pixel.show();

      delay(wait);

      for (int i=0; i &lt; pixel.numPixels(); i=i+3) {
        pixel.setPixelColor(i+q, 0);        //turn every third pixel off
      }
    }
  }
}

//Theatre-style crawling lights with rainbow effect
void theaterChaseRainbow(uint8_t wait) {
  for (int j=0; j &lt; 256; j++) {     // cycle all 256 colors in the wheel
    for (int q=0; q &lt; 3; q++) {
      for (int i=0; i &lt; pixel.numPixels(); i=i+3) {
        pixel.setPixelColor(i+q, Wheel( (i+j) % 255));    //turn every third pixel
on
      }
      pixel.show();

      delay(wait);

      for (int i=0; i &lt; pixel.numPixels(); i=i+3) {
        pixel.setPixelColor(i+q, 0);        //turn every third pixel off
      }
    }
  }
}

// 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) {
  WheelPos = 255 - WheelPos;
  if(WheelPos &lt; 85) {
    return pixel.Color(255 - WheelPos * 3, 0, WheelPos * 3);
  }
  if(WheelPos &lt; 170) {
    WheelPos -= 85;
    return pixel.Color(0, WheelPos * 3, 255 - WheelPos * 3);
  }
  WheelPos -= 170;
  return pixel.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
}
```

3D Printing

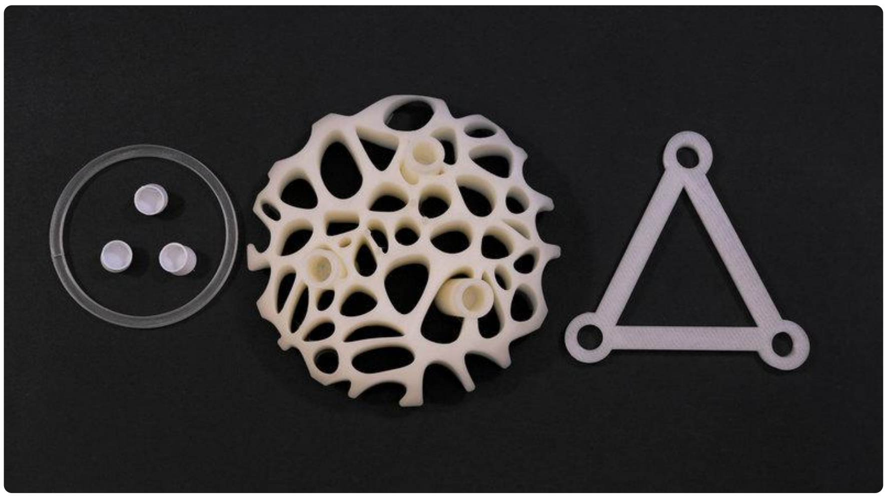
Download and 3D Print

The 3D printed parts can be downloaded with the link below.

Thingiverse

Youmagine

Pinshape



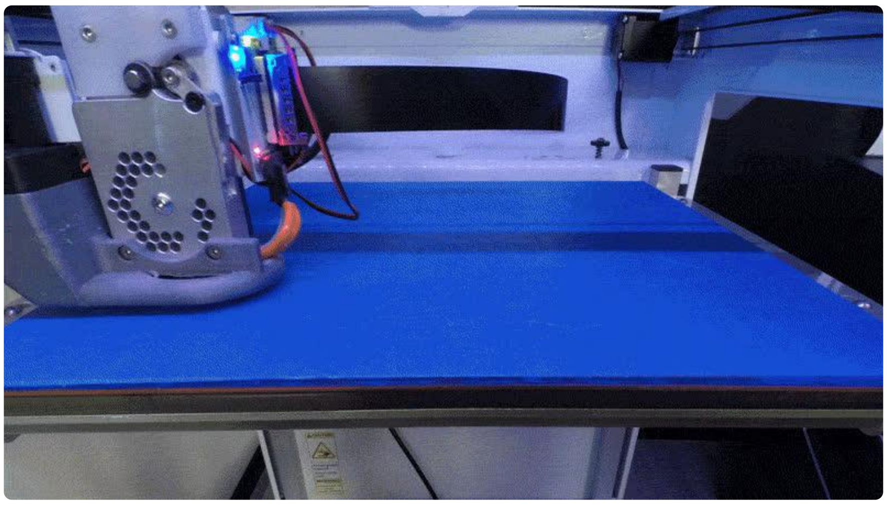
Edit Design

Materials & Slice Settings

This design requires a 12 x 12 inch bed size. If your 3D printer isn't capabile of larger sizes, you could print the pieces separately and glue them together.

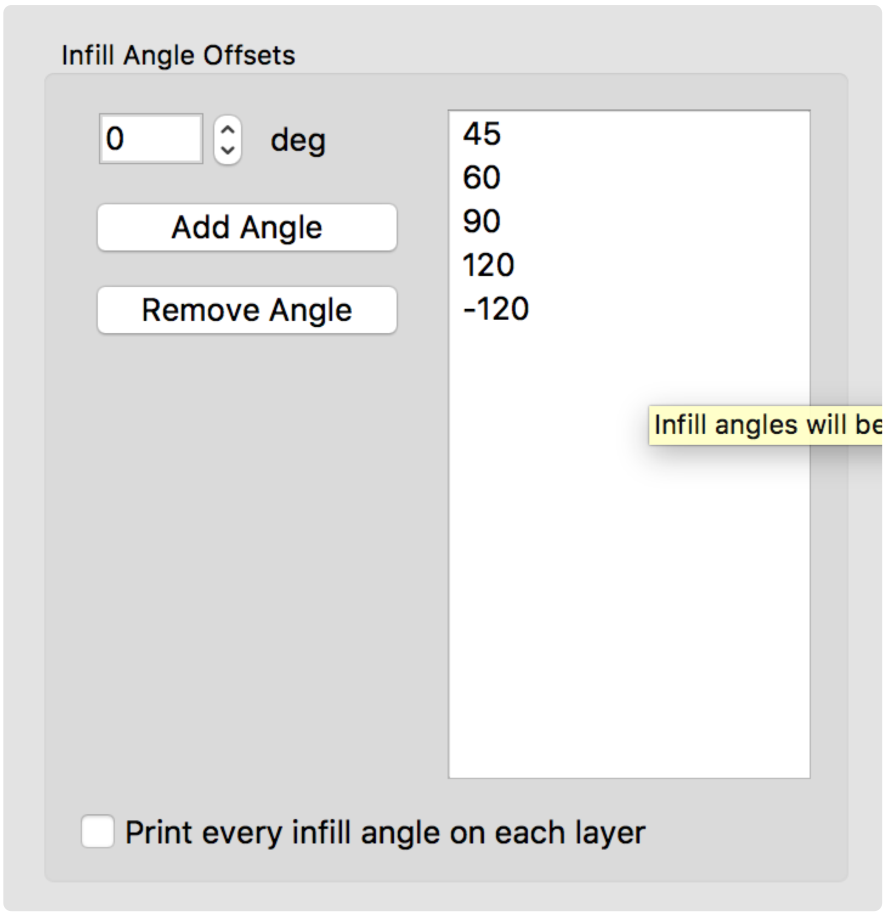
The stool legs use TPU material, so will need an extruder capable of printing flexable filament. Reference the slice settings in the table below – We're using Simplify3D to slice the parts.

stool-seat.stl stool-frame.stl stool-feet.stl stool-ring.stl	PLA 245c / 60c bed No supports 30% infill 2mm Z-Hop 90mm/s print speed 120mm/s travel speed Infill angle offsets: 45, 60, 90, 120, -120	We used a 6mm brim to insure the bottom doesn't curl up while printing. The seat part take about 30 hours to complete. Z-Hop is used to prevent any collisions with the print head and any edges that might curl up. We used infill angle offsets to increase the strength of the seat. The stool feet are printed in original Ninjaflex to make them grippy, but you could also use semi-flex or cheetah.



Increase infill strength

We utilized the 3d infill technique by cetting the infill angle offsets to 45, 60, 90, 120, -120. This will alternate how the infill is stacked on top of each other, giving the seat more strength.


Infill Angle Offsets

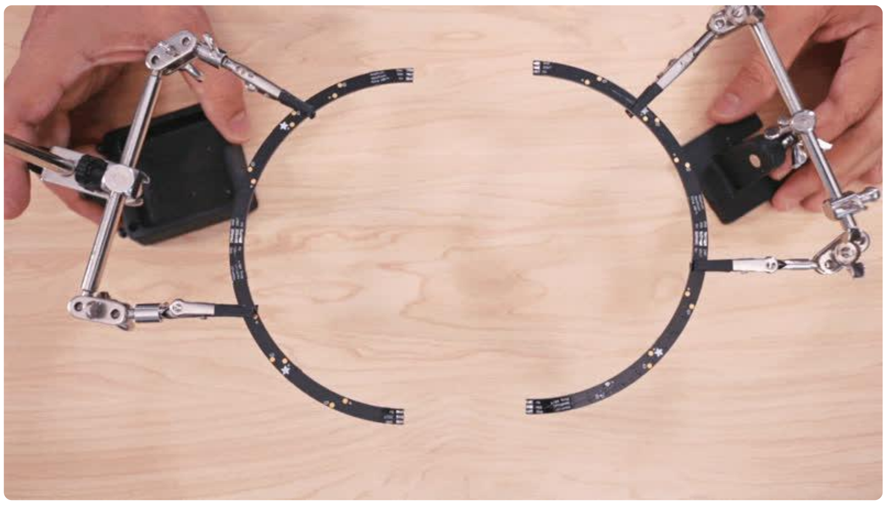
0 deg

45
60
90
120
-120

Infill angles will be

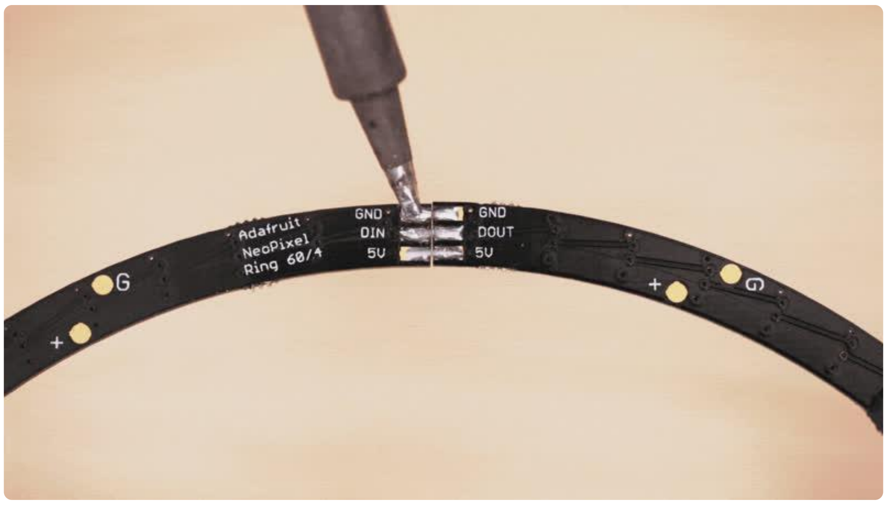
☐ Print every infill angle on each layer

Assembly



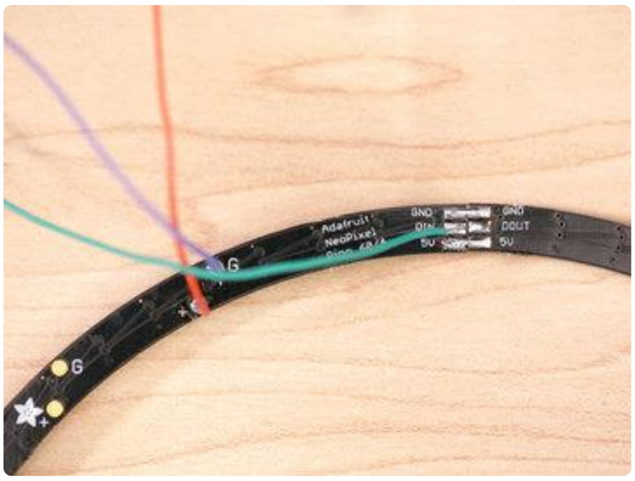
Assemble 60 NeoPixel Ring

The 60 ring neopixel rings come in quarter sized sections that will need to be soldered to each other to make the full circle. We recommend using multiple third helping hands to hold the four pieced together while soldering.



Connect 1/4 NeoPixel PCBs

To solder the quarter pieces together, start by tinning each side and then carefully slide the solder from one side of the pad to the other, making sure both pads are connected. It will take some practice, but you'll quickly get a hang of it.

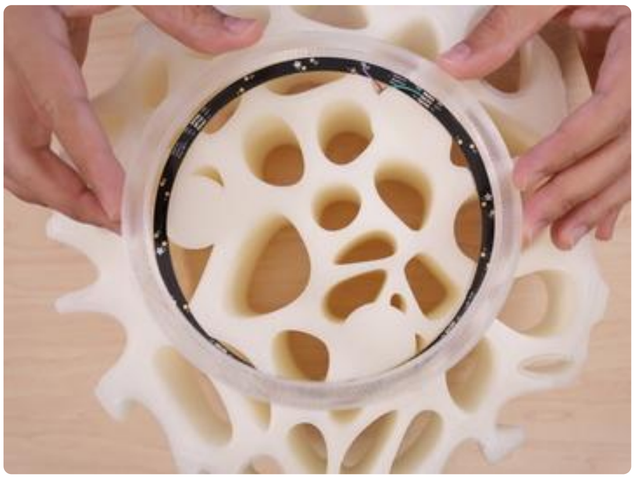


Solder Wires

Solder all of the sides together except the last DIN and DOUT, leave those two last ones separated. DIN is where the data will feed into the neopixel ring.

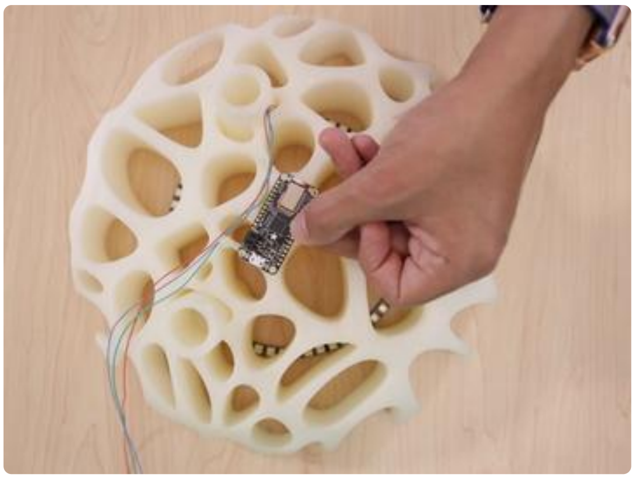
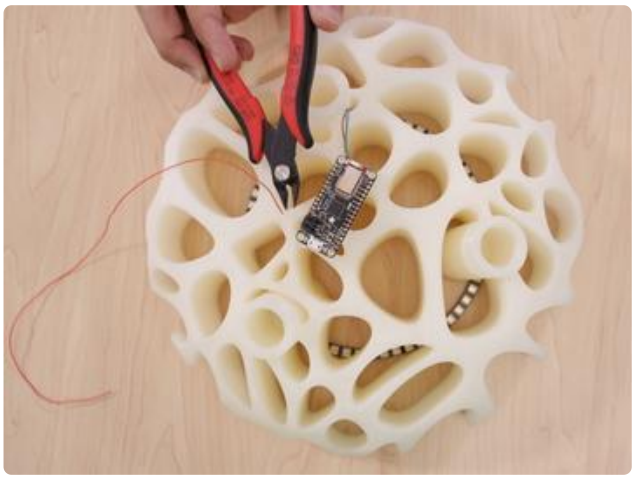
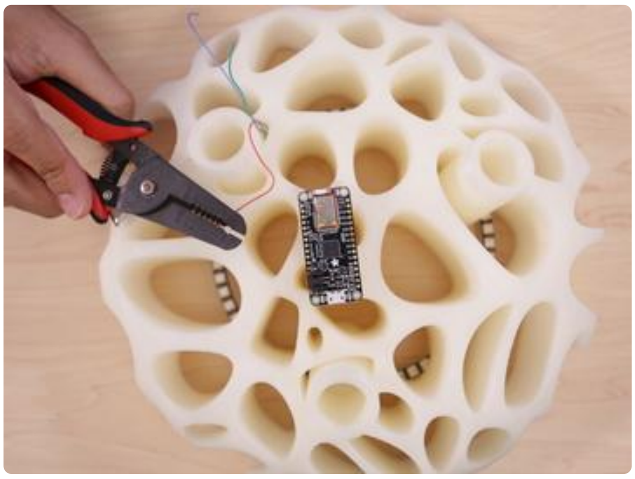
Thread wires to the other side

We can use to small opening near the mounts on the bottom side of the seat to thread the wire into.



Ring Cover

Place the ring face down so the lights can diffuse the seat from the top down. The cover snaps on top to protect and secure the ring in place.

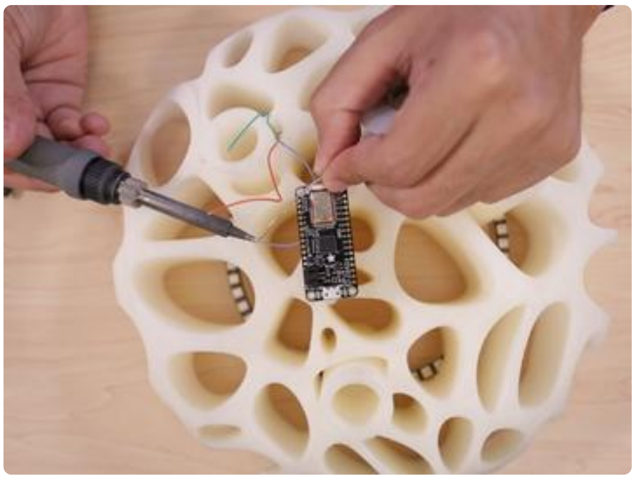
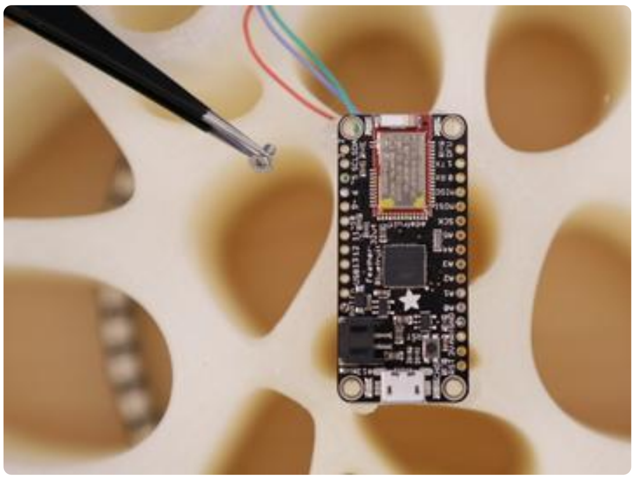
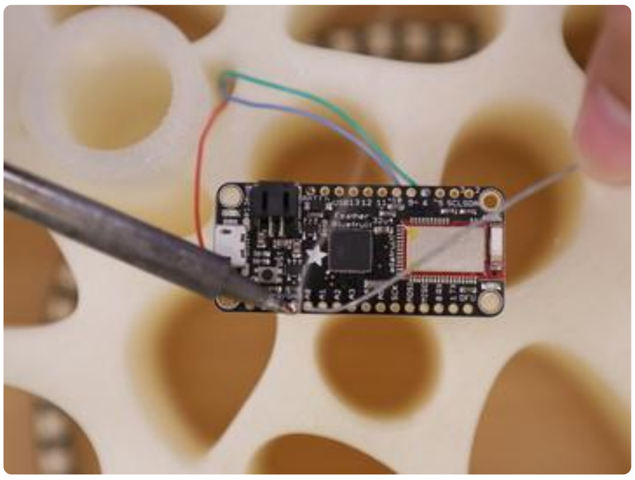




Measure and Cut Wires

Place the Feather board on the mounts and then measure wires for power, ground and data to connect the neopixel ring.

Use flush cutters to trim the wires with a little bit of slack, you can always recut them if they are too long.

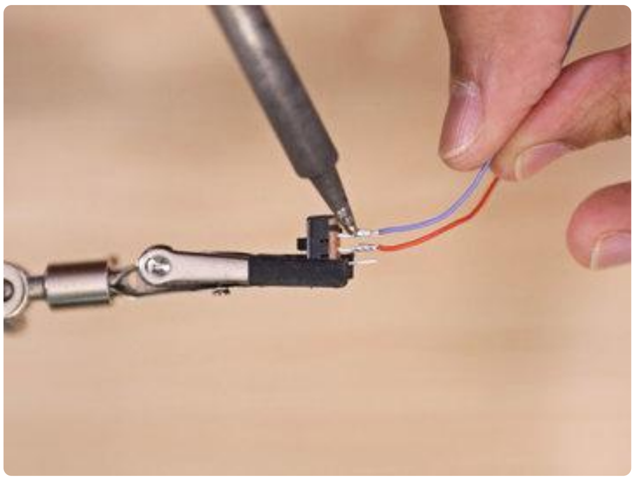
Strip each wire to prepared them for tinning.

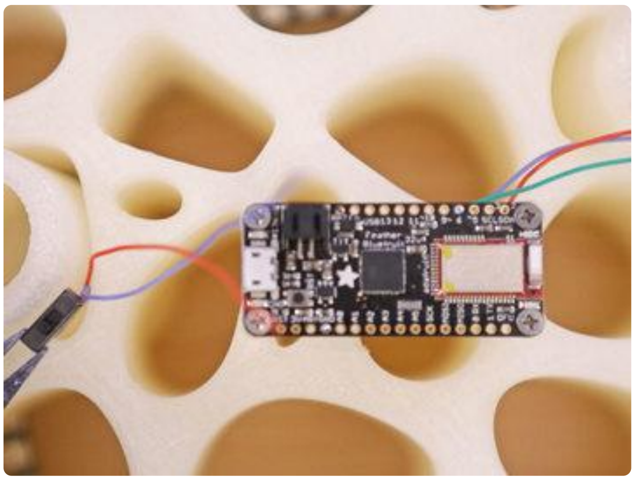




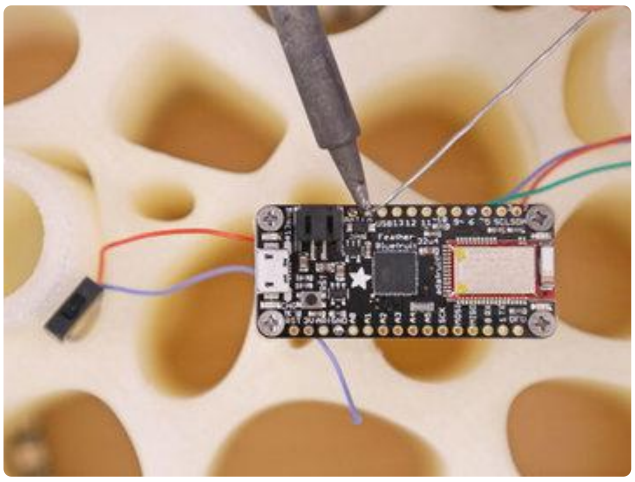
Tin, Solder and Screws

2.5mm screws are use to securely hold the feather board on the four mounts. We recommend tapping each mount before securing the board on top.

Now we can tin, solder and solder each wire to the board. Connect power on the neopixel ring to 3v on the Feather board, Ground to G and DIN to Pin 6 on the Feather board.





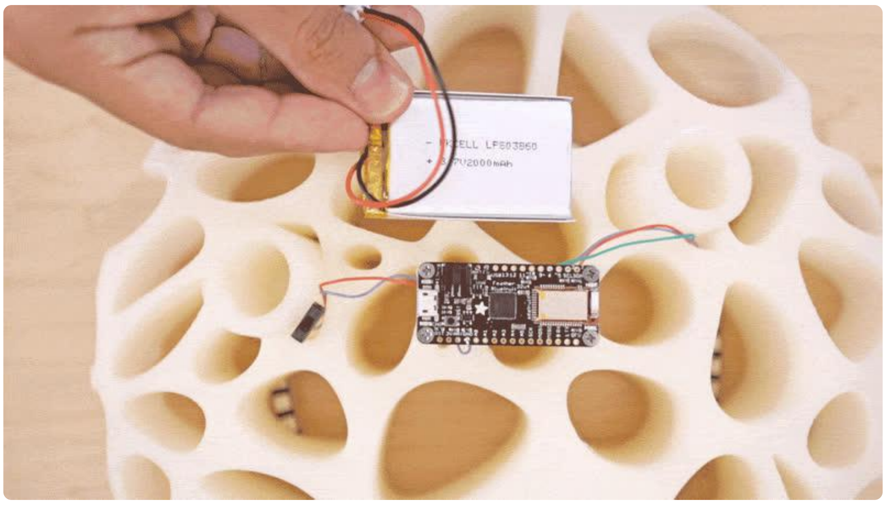


Slide Switch

To easily power the circuit on and off we'll use a slide switch. We'll get the switch ready by tinning two of the leads and soldering a wire that will connect to the EN pins and the Ground pin.

Since this will be the second wire soldered to the ground pin, it's easier to solder this wire to the top.

Tweezers help to maneuver wires into place.

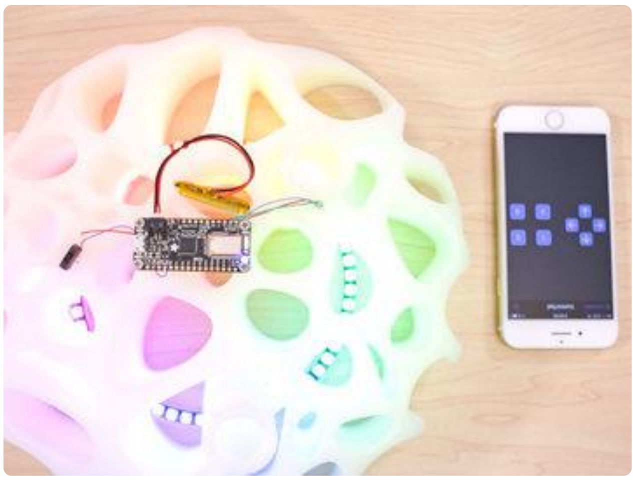


Battery

To power the circuit we can fit a 2,000mAh battery in the slot close to the board. It should have a tight fit, but we reccomend using double stick foam tape to secure it place.

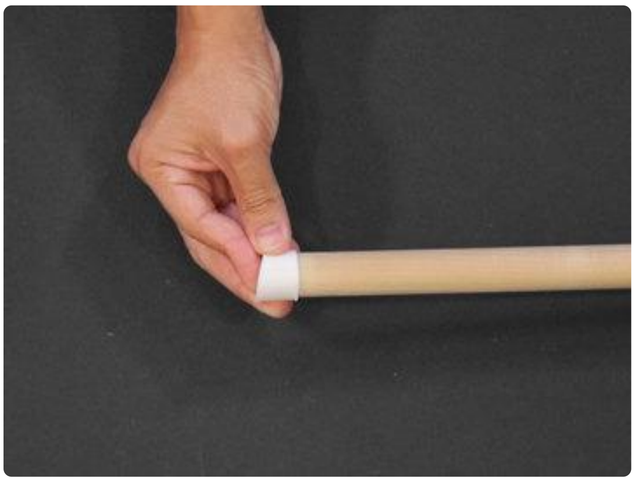
Load Sketch

Plug the micro USB cable into board and your computer to load the sketch.



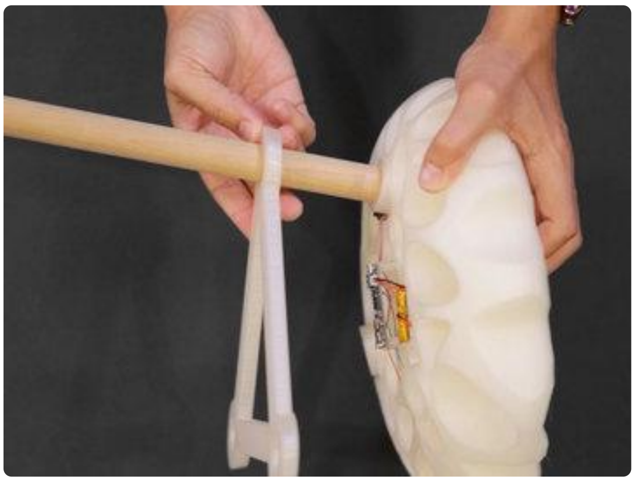
Make sure to have a data cable and not a power only USB cable.

The Adafruit Bluefruit app connects to the feather ble board and works on iOS and Android devices to update colors and animations on the LEDs. You can find them on the Apple App Store or Google Play Store.



Rubber Feet

Three 7/8 inch dowels use rubber ninja flex feet to keep them from digging into soft surfaces and from slipping on hard surfaces.



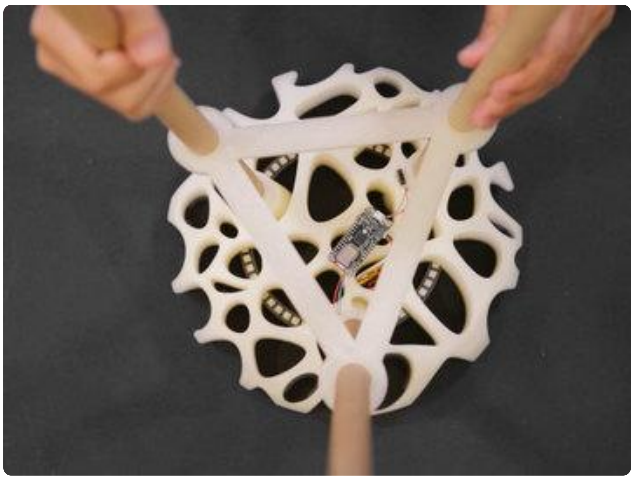
Frame

Align the triangle support frame to the three slots on the seat and then fit each leg through the frame and into each slot.

Level Legs

If the tolerances are a little loose you can use making tape on the ends to get a tight fit in the slots.

Make sure all of the legs are level and you're ready to sit!



Customizing and printing your own furniture is a lot of fun, but adding electronics is really rewarding. You can build on top of this design to make a seat that can change colors based on weather or even flash when you receive different alerts from twitter or github repos.