No-Solder PaperCraft Crystal Light Strand

Created by Erin St Blaine

https://learn.adafruit.com/no-solder-papercraft-crystal-light-strand

Last updated on 2024-03-08 03:46:40 PM EST

Table of Contents

Overview

Crystals catch our eye and our imagination like no other substance on earth. These unique structures are sought out and prized in almost every culture. Spiritualists and scientists alike have a fascination with crystals -- the way they catch the light is unlike anything else on earth. They are rare and beautiful, and they resonate with us on so many levels.

These paper crafted crystals are a remarkable facsimile of the real thing. Their iridescence and faceted "occlusions" evoke a feeling of deep caverns, mystical seers and fortresses of solitude in my imagination. I just can't stop making them! They are oh so satisfying to fold, and oh so satisfying to hold.

Add a strand of twinkly NeoPixels, and we've got a perfect recipe for magic.

What Are They Made Of?

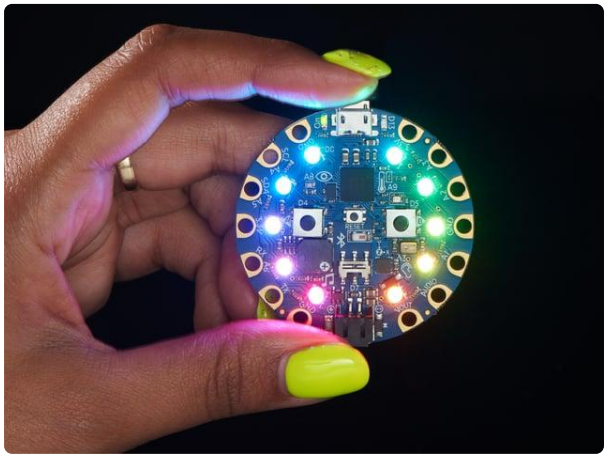
These crystals are made from iridescent cellophane wrap (from the craft store, like you find in fancy gift baskets) that has been crinkled up and then run through a laminating machine. They can be cut by hand with a utility knife, or cut out on a vinyl cutting machine like a Cricut or Silhouette.

Then they are folded into shape, tab-in-slot style, and finally we add a small piece of iridescent cellophane inside to mimic the crystalline occlusions. This is surprisingly convincing -- they look like the real thing, but are light and flexible -- perfect for cosplay applications or just stringing up around the house.

The crystal patterns in this guide are designed to slot perfectly around Adafruit's NeoPixel Dot strand. Each strand has 20 lights, and we've included crystal patterns in varying shapes and sizes, so you can customize the look of your strand.

Or, you can order pre-cut crystal gems kits here. (https://adafru.it/OMF)

Adafruit Parts Needed



Circuit Playground Bluefruit - Bluetooth Low Energy

Circuit Playground Bluefruit is our third board in the Circuit Playground series, another step towards a perfect introduction to electronics and programming. We've...

https://www.adafruit.com/product/4333

The NeoPixel dots come in 2" or 4" spacing. For home decor I prefer the 4" strand, but the 2" strand might suit your purposes better if you're doing a costume or other art project. Each strand has 20 lights. The 4" strand is just over 9 feet long, so if you want to decorate a whole room you may want at least 2-3 strands. They come with connectors already attached, so daisy-chaining them is very simple.

Adafruit NeoPixel LED Dots Strand - 20 LED 4" Pitch

Attaching NeoPixel strips to your costume can be a struggle as the flexible PCBs can crack when bent too much. So how to add little dots of color? Use these stranded NeoPixel dots!...

https://www.adafruit.com/product/3631

Adafruit NeoPixel LED Dots Strand - 20 LEDs at 2" Pitch

Attaching NeoPixel strips to your costume can be a struggle as the flexible PCBs can crack when bent too much. So how to add little dots of color? Use these stranded NeoPixel dots!...

https://www.adafruit.com/product/3630

You can simply solder the lights to the Circuit Playground if you've got soldering chops. Check out this Make It Glow - How To Solder NeoPixels (https://adafru.it/LDV) guide if you want to pick up this very useful skill.

However, for this project you don't need to be able to solder. This bolt-on kit will securely attach the LEDs to the board with no soldering required.

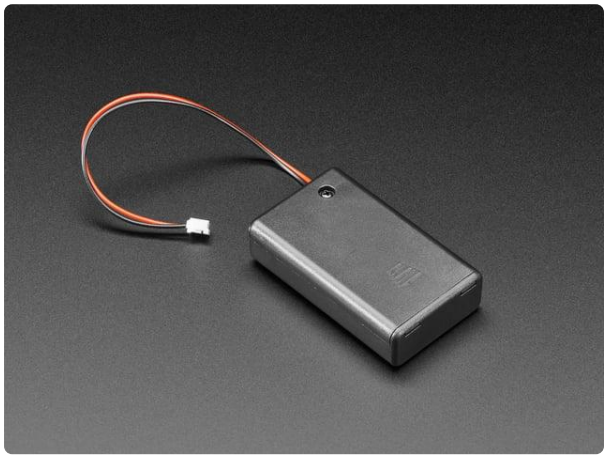
Bolt-On Kit for Circuit Playground, micro:bit, Flora or Gemma

You have a Circuit Playground Express, and want to connect some wires to it for adding LEDs or sensors or speakers? You can use our...

https://www.adafruit.com/product/4103

If you're planning to plug these into a wall socket, you can do that with a USB cable (http://adafru.it/4148) and USB wall wart. If you want to take it on-the-go as part

of a costume or mobile project, you'll need a battery pack. I like this one since it already has an on/off switch included.



3 x AAA Battery Holder with On/Off Switch and 2-Pin JST
This battery holder connects 3 AAA batteries together in series for powering all kinds of projects. We spec'd these out because the box is slim, and 3 AAA's add up to about...
https://www.adafruit.com/product/727

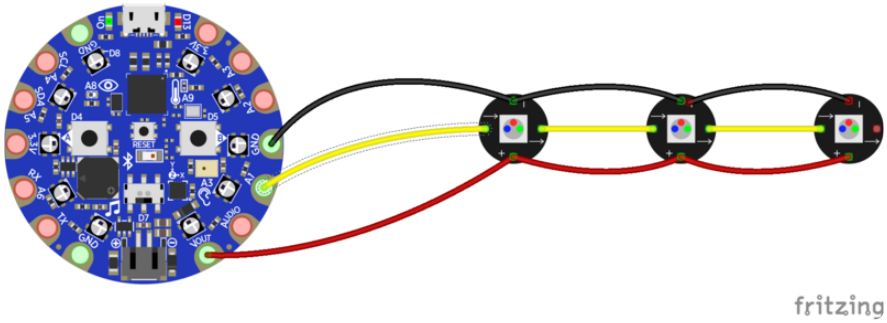
A nice set of wire strippers always save you frustration.

Hakko Professional Quality 20-30 AWG Wire Strippers
These are the finest wire strippers we have used, and if you have to do a lot of wiring, you will agree! They have soft rounded grips - very comfortable to use, and precision ground...
https://www.adafruit.com/product/527

Additional Materials & Tools

- **Iridescent cellophane wrap** -- This is the stuff you use to make your gift bags look fancy. It comes in a roll like wrapping paper, in a variety of colors. I like "opal" but there are lots of colors to choose from. Check your local craft store or shop online. Just be sure you're getting iridescent! Plain transparent or colored cellophane will not give the same look.
- **Laminating machine** -- you can find them for around $25 on Amazon, and once you've got one, you'll laminate EVERYTHING.
- **3 mil laminating pouches** -- these come in packs of 50 or more, and I'd recommend getting at least that many. You can cut 1-2 crystals per pouch, depending on how big you make them, but get enough to account for errors.
- **Vinyl Cutting Machine** like a Cricut or Silhouette, OR a **sharp utility knife** and lots of elbow grease

Circuit Diagram



The NeoPixel strand connects as follows:

- Red wire --> **VOUT**
- Middle wire --> **A1**
- Remaining Wire --> **GND**

Be sure you're connecting to the IN end of the strip. You can connect power (**VOUT**) and ground (**GND**) at either end, but the middle data connection MUST connect to the correct end of the strip. Check for an arrow on the back of the pixels to be sure you've got the right end. More about this in the assembly section!

Power your lights with a USB cable via the USB port, or a battery pack connected to the JST port.

Code with MakeCode

MakeCode is an easy way to get up and running with the Circuit Playground. No prior coding knowledge is needed, and it's an easy way to experiment and learn to think like a coder. You just drag and drop code blocks, like building with Lego. MakeCode makes coding fun!

Head over to this Intro to MakeCode (https://adafru.it/AEp) guide for more info on getting started with MakeCode.

How To Upload the Code

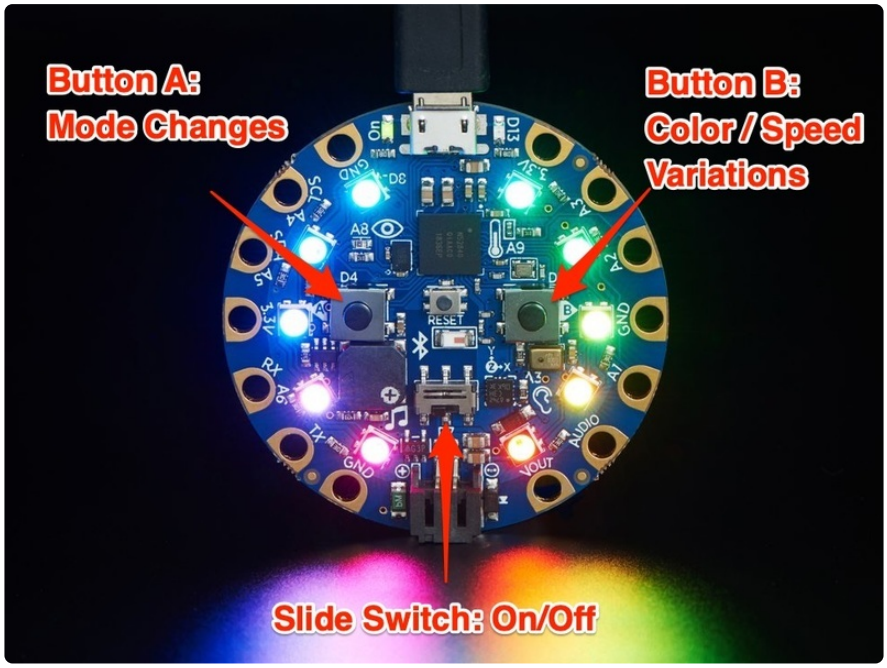
Open the project code in MakeCode

https://adafru.it/LAs

1. Open the project with the button above and click the blue **DOWNLOAD** button near the bottom of the project.
2. Plug your Circuit Playground into your computer via its USB port and click the "reset" button. All the lights will turn green and your Circuit Playground will appear as a drive on your computer called **CPLAYBOOT**.
3. Drag the code you just download onto this drive to program the Circuit Playground -- like putting files on a USB stick. That's all you need to do.

Note: If you plug in the board and you see a drive called **CIRCUITPY** appear, press the reset button again (double-click) to get to **CPLAYBOOT**.

How To Use It



On/off:

The tiny slide switch on the face of the Circuit Playground turns the light strand on or off. Slide right for on, slide left for off. (So, if you just downloaded the code and you don't see anything happening, flip this switch! You might just be in "off" mode.)

Modes

This code has four different modes. Click **button A** to cycle between them:

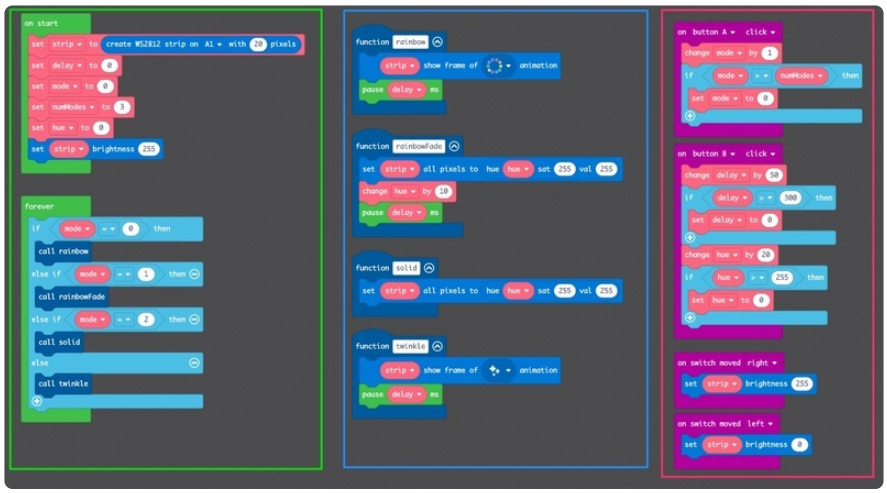
1. Rainbow: an animated LED rainbow. Who doesn't love rainbows?
2. Rainbow Fade: the lights are all the same color and fade through the rainbow as a unit
3. Solid: Pick a color, any color ya want!
4. Twinkle: Lovely glittery white twinkles

Variations

Button B is our variation button. Click it to vary the **speed** of the rainbow and twinkle animations, and the **color (hue)** of the solid animation. There are seven speed levels, from fast to slow, and the hue will just keep changing and rotating each time you press the button.

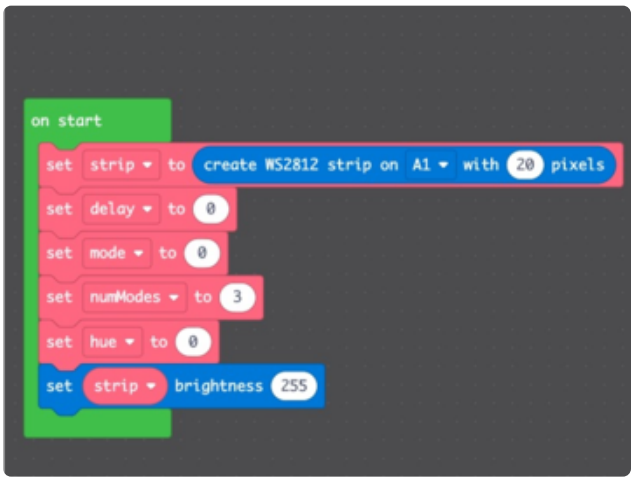
How to Customize Your Code

You can add your own modes, give yourself more speed variations, or set it to start up on your favorite mode. The possibilities are endless! Poke around and change some stuff, and if it stops working you can always come back here and start fresh. Make it yours.



The green box on the left contains two blocks: `on start` and `forever`. These are found under the **LOOPS** tab at the left. Anything in the `on start` block runs just once, when the board first powers up. Anything in the `forever` loop runs over and over, forever.

On Start Block



The first line sets up our light strand. If you have more than one strand daisy-chained, change the `set strip to create WS2812 strip` number to reflect your total number of lights.

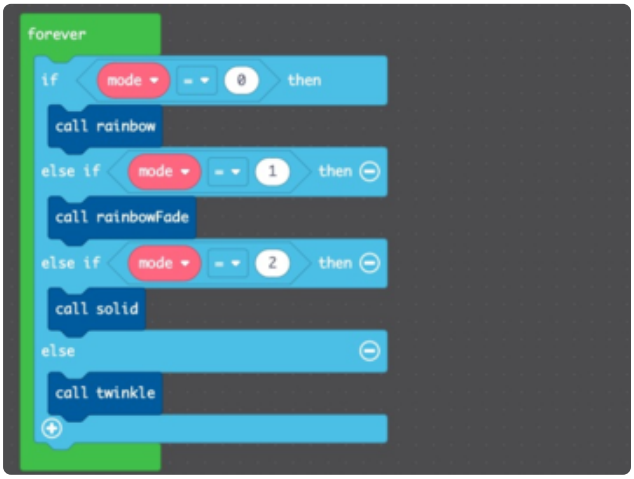
Next, we define our variables.

If you add or remove modes, change `numModes` to reflect the total number of modes (including mode 0).

To select a different start-up mode, change `set mode to 0` to the number of the mode you want to appear first.

You can also change the starting `hue` by choosing a number between 0 and 255.

Forever Block

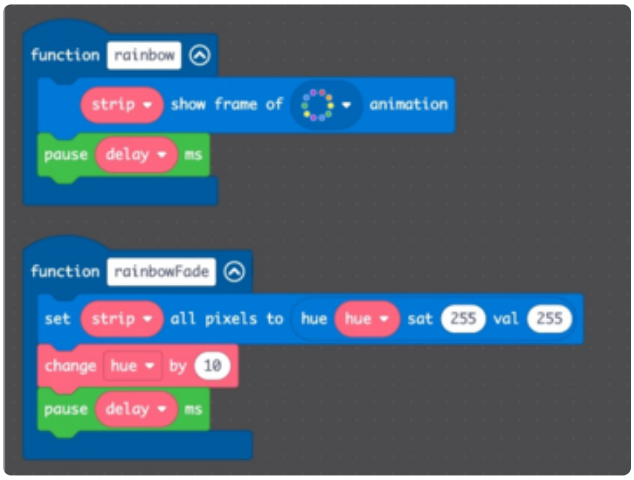


Here is where we set up the order of our modes. If you want to add more modes you can do it here by clicking the + button at the bottom and calling more functions. Be sure to increment the mode number in each block you make, and don't forget to change `numModes` in the previous block if you add or subtract modes.

I really like having `solid` come after `rainbowFade`. I can choose which solid color to "land" on by the timing of when I press **button B** -- as the lights fade through the rainbow, press the button to "capture" that color as the solid hue.

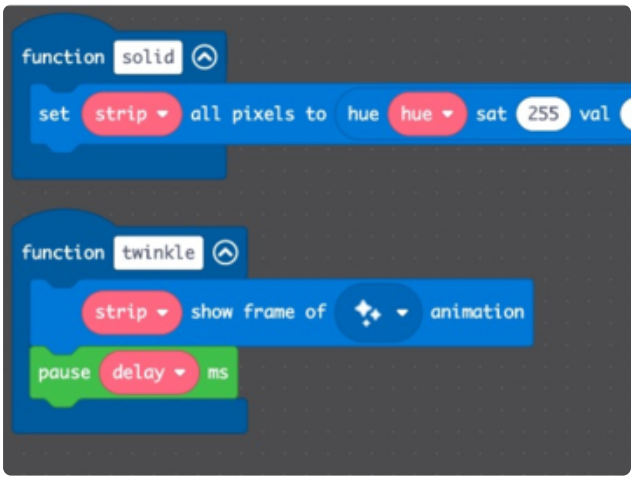
Functions

The blue box in the middle contains our functions. These are the actual LED modes themselves, which we call in the forever loop. You can add as many functions as you like.



The `rainbow` function is using MakeCode's built-in rainbow animation. We're using `show frame of animation` instead of `show animation` because that allows us to change the speed with the `delay` variable, by pressing **button B**.

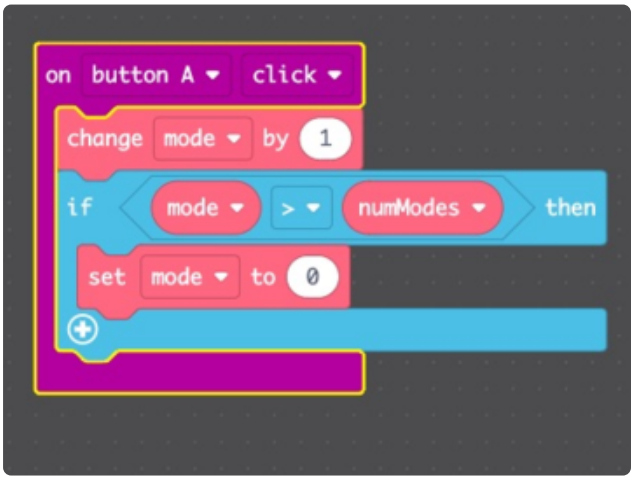
The `rainbowFade` function can be customized too. To make the lights more pastel / less saturated, change the `saturation` number: 255 is fully saturated, and 0 is pure white. To make the fade go more slowly / smoothly, decrease `change hue by` to a smaller number. Set this to `1` for a very slow, smooth fade -- experiment to see what you like.



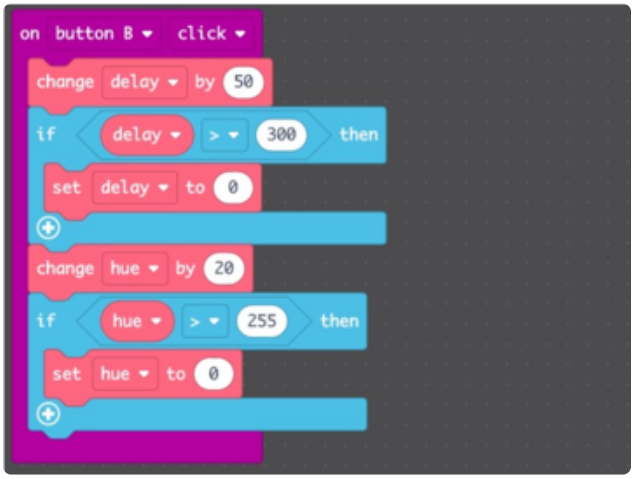
The `solid` function is a simplified version of `rainbowFade`. You can vary the saturation here too if you'd like.

The `twinkle` function is another built-in animation provided by MakeCode. There are 6 animations in that dropdown -- experiment with the others to see if there's one you like!

Button Controls



This block is where we tell the CircuitPlayground to scroll through the modes each time **button A** is clicked. When it reaches the last mode (defined by the `numModes` variable from the `on start` block) it resets to 0.



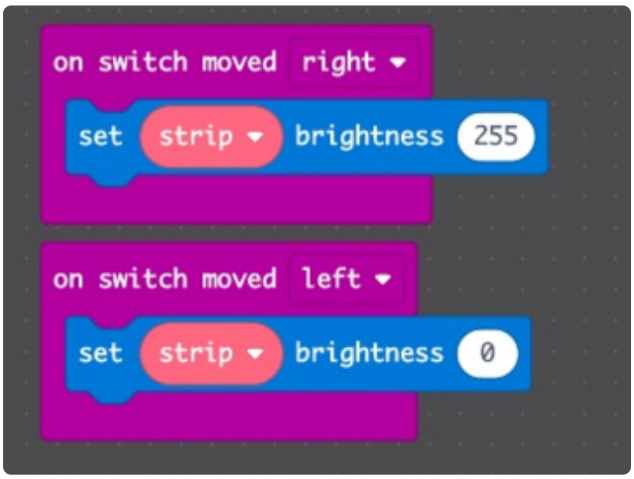
Button B is where we've created our variations. I'm varying speed and color, but you could also vary brightness or saturation or any other variable you can dream up.

Try changing the numbers around to customize how many speed levels there are or how much the color changes each time the button is pressed.

There are also other triggers available -- you can trigger a mode change or variation when you shake or move the CircuitPlayground, or whenever a loud sound is heard, or a whole host of other options. Look under the INPUT tab to see them all.

This is an easy way to make your project react to its environment. Make the crystals change color whenever you clap your hands or jump up and down!

On/Off Switch



Here's where we turn the lights on and off with the slide switch. We've done this by changing the brightness to 0 or back to 255, so if you want to change the overall strand brightness be sure to change it here as well as in on start.

Doing it this way means the Circuit Playground will "remember" the light mode and variation you were in when you turn the lights back on. Cool!

However, remember that the whole project is NOT turned off when the slide switch is in the off position. The board is still awake and using power, so if you've got a battery powered project, this mode will slowly drain your battery. Turn it off with the switch on the battery pack or unplug your battery when you're ready to leave it off long-term.

Troubleshooting

If you are having trouble, here are a few things to check:

- Is your slide switch in the "off" position? If you download the code and nothing happens, flip the switch the other way to see if that fixes things.
- Are you connected to pin **A1** with your data wire? Double check your wiring to be sure it matches the diagram.
- Did you connect to the IN end of the strip? The lights should be face up when the Circuit Playground is face up, without any twist in the wires. If they're face down, you've connected to the wrong end of the strip. Switch it around and try again.
- Are you using a Circuit Playground Express instead of a Circuit Playground Bluefruit? This code will work with the Express, but you need to select it in the MakeCode window. Click the `...` button next to the `Download` button and select **Choose Hardware** to switch boards.

If you still can't get it working, head over to the Circuit Playground BlueFruit Intro Guide (https://adafru.it/GYc) for more help and suggestions.

Code with CircuitPython

CircuitPython is a fast growing programming platform based on Python, that's easy to learn and customize. This project makes use of the Circuit Python LED Animations (https://adafru.it/LZF) code by Kattni Rembor (https://adafru.it/L-a) to quickly and easily add gorgeous animations to your light strand. There are more pre-made animations available than I'm using for this guide, so go check it out if you need more modes on your light strand.

How To Upload the Code

Download the Latest Version of CircuitPython for Circuit Playground Bluefruit

https://adafru.it/FNK

Step 1: Update CircuitPython

1. Download the latest version of the Circuit Python operating system using the button above.
2. Plug your Circuit Playground into your computer via its USB port and double-click the "reset" button. All the lights will turn green and your Circuit Playground will appear as a drive on your computer called **CPLAYBOOT**.
3. Drag the code you just download onto this drive to install CircuitPython -- like putting files on a USB stick.

Note: If you plug in the board and you see a drive called **CIRCUITPY** appear, press the reset button again (double-click) to get to **CPLAYBOOT**.

Download CircuitPython Library Bundle

https://adafru.it/ENC

Step 2: Install the Required Libraries

1. Click the button above to download the latest CircuitPython Library Bundle Release.

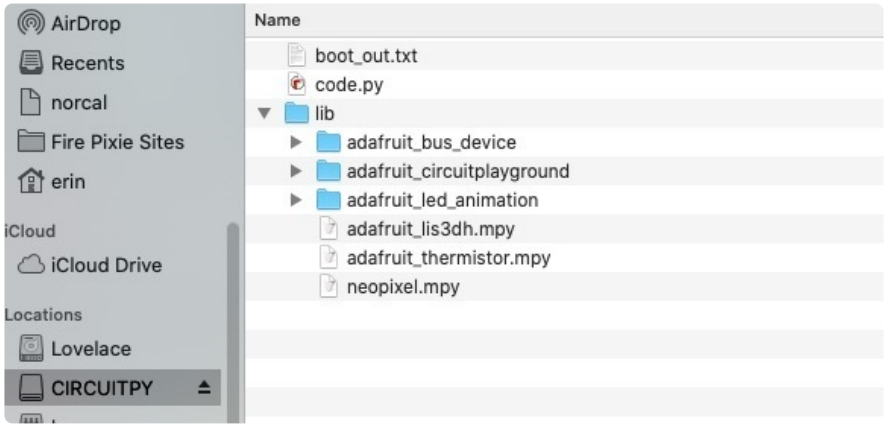
2. Open the file you just downloaded and look in the **lib** folder. Find these files and copy them into the **lib** folder on your Circuit Playground's **CIRCUITPY** drive.

- **adafruit_bus_device**
- **adafruit_circuitplayground**
- **adafruit_led_animation**
- **adafruit_lis3dh.mpy**
- **adafruit_thermistor.mpy**
- **neopixel.mpy**

Step 3: Upload the Code

Copy the code from the code window below and save it as **code.py** at the root of your **CIRCUITPY** drive.

Your **CIRCUITPY** drive should look like this when you're done:

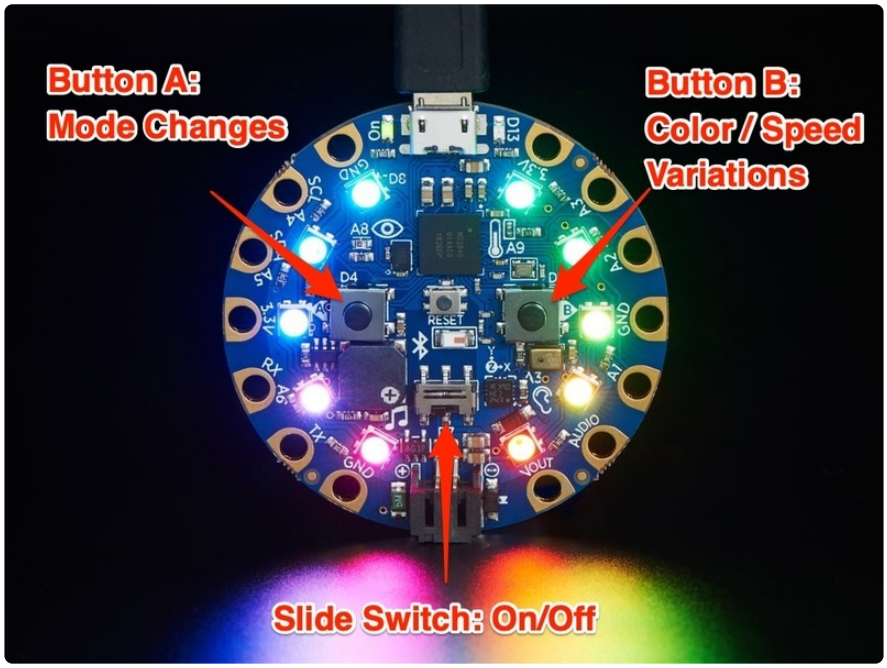



```
# SPDX-FileCopyrightText: 2020 Erin St Blaine for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""
Crystal Gem Light Strand Project By Erin St Blaine for Adafruit Industries
https://learn.adafruit.com/no-solder-papercraft-crystal-light-strand

Circuit Playground Bluetooth with LED strand attached runs 4 different variable
animation modes.

Code by Rose Hooper using Adafruit's LED Animation Library:
 https://learn.adafruit.com/circuitpython-led-animations/overview
"""
# pylint: disable=attribute-defined-outside-init

#Import libraries
import board
import neopixel
from rainbowio import colorwheel
from adafruit_circuitplayground import cp
from adafruit_led_animation.animation.solid import Solid
from adafruit_led_animation.animation import Animation
```

```
from adafruit_led_animation.animation.rainbow import Rainbow
from adafruit_led_animation.animation.sparkle import Sparkle
from adafruit_led_animation.sequence import AnimationSequence
from adafruit_led_animation.color import WHITE

speeds = (0.25, 0.125, 0.1, 0.08, 0.05, 0.02, 0.01)  # Customize speed levels here
# periods = (7, 6, 5, 4, 3, 2, 1)

class RainbowFade(Animation):

    _color_index = 0
    def __init__(self, pixel_object, speed, name):
        super().__init__(pixel_object, speed=speed, color=WHITE, name=name)

    def draw(self):
        self.color = colorwheel(self._color_index)
        self._color_index = (self._color_index + 1) % 256
        self.fill(self.color)


current_speed = 4

# Set your number of pixels here
pixel_num = 20

pixel_pin = board.A1
pixels = neopixel.NeoPixel(pixel_pin, pixel_num, brightness=1, auto_write=False)

# Animation Setup

rainbow = Rainbow(pixels, speed=speeds[current_speed], period=2, name="rainbow",
step=3)
sparkle = Sparkle(pixels, speed=speeds[current_speed], color=WHITE, name="sparkle")
rainbowfade = RainbowFade(pixels, speed=speeds[current_speed], name="rainbowfade")
solid = Solid(pixels, color=colorwheel(0), name="solid")

# Animation Sequence Playlist -- rearrange to change the order of animations

animations = AnimationSequence(
    rainbow,
    rainbowfade,
    solid,
    sparkle,
    auto_clear=True,
    auto_reset=True,
)

solid.speed = 0.01
solid_color = 0

while True:
    if cp.switch:  # if slide switch is in the "on" position, run animations
        animations.animate()  #play animation sequence
        if cp.button_a:
            animations.next()
            while cp.button_a:
                continue

        if cp.button_b:
            if animations.current_animation.name == "solid":
                solid_color = (solid_color + 8) % 256
                animations.current_animation.color = colorwheel(solid_color)
            else:
                current_speed += 1
                if current_speed >= len(speeds):
                    current_speed = 0
                rainbow.speed = speeds[current_speed]
                sparkle.speed = speeds[current_speed]
                rainbowfade.speed = speeds[current_speed]
```

```
                print(speeds[current_speed])
            while cp.button_b:
                continue
    else:  # If slide switch is in the "off" position, set pixels to black
        pixels.fill(0)
        pixels.show()
```

How It Works



On/off:

The tiny slide switch on the face of the Circuit Playground turns the light strand on or off. Slide right for on, slide left for off. (So, if you just downloaded the code and you don't see anything happening, flip this switch! You might just be in "off" mode.)

Modes

This code has four different modes. Click **button A** to cycle between them:

1. Rainbow: an animated LED rainbow. Who doesn't love rainbows?
2. Rainbow Fade: the lights are all the same color and fade through the rainbow as a unit
3. Solid: Pick a color, any color you want!
4. Twinkle: Lovely glittery white twinkles

Variations

Button B is our variation button. Click it to vary the **speed** of the rainbow and twinkle animations, and the **color (hue)** of the solid animation. There are seven speed levels, from fast to slow, and the hue will just keep changing and rotating each time you press the button.

How to Customize Your Code

You can add your own modes, give yourself more speed variations, or set it to start up on your favorite mode. The possibilities are endless! Poke around and change some stuff, and if it stops working you can always come back here and start fresh. Make it Yours.

This code is written for one 20-pixel light strand. Here are a few customization hints.

```
current_speed = 4

# Set your number of pixels here
pixel_num = 20
pixel_pin = board.A1
pixels = neopixel.NeoPixel(pixel_pin, pixel_num, brightness=1, auto_write=False)
```

Change `current_speed` to change the initial startup animation speed level.

Change `pixel_num` to match the number of pixels in your strip.

Change `brightness=1` to anything between 0.0 and 1.0 to change the overall master brightness of your strip.

Animations

```
# Animation Setup

rainbow = Rainbow(pixels, speed=speeds[current_speed], period=10, name="rainbow",
step=3)
sparkle = Sparkle(pixels, speed=speeds[current_speed], color=WHITE, name="sparkle")
rainbowfade = RainbowFade(pixels, speed=speeds[current_speed], name="rainbowfade")
solid = Solid(pixels, color=colorwheel(0), name="solid")

# Animation Sequence Playlist -- rearrange to change the order of animations

animations = AnimationSequence(
    rainbow,
    rainbowfade,
    solid,
    sparkle,
    auto_clear=True,
```

```
    auto_reset=True,
)
```

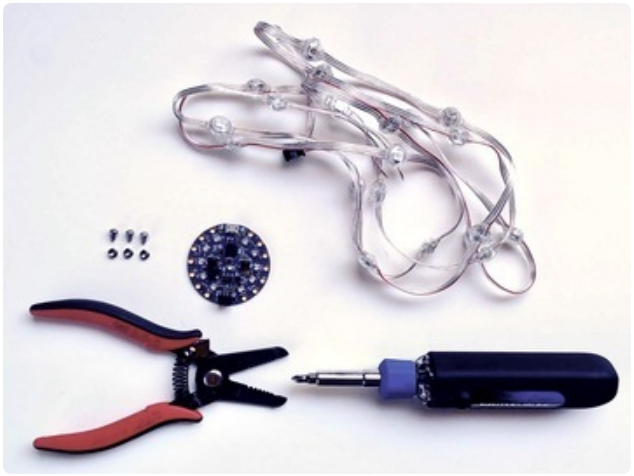
You can add more animations or re-order them however you'd like. Check out the CircuitPython LED Animations Guide (https://adafru.it/LZF) to see what other animations this library offers. There are several different Rainbow animations, Pulse, Chase, and some Sparkle variations that are a lot of fun to play with.

You'll need to change the code in three places:

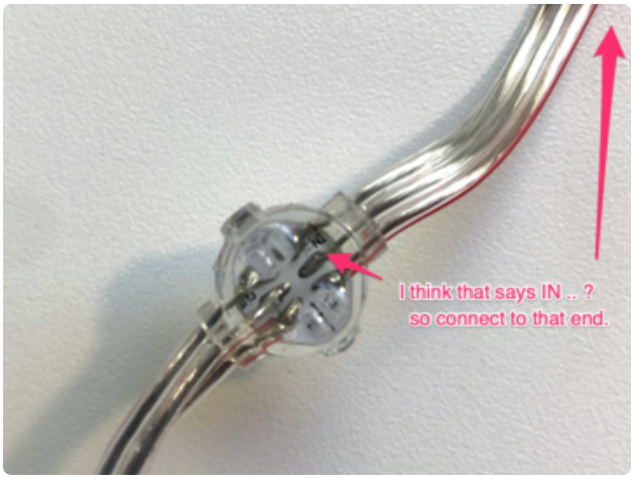
1. Import the animation at the very top
2. Create a line for each one under #Animation Setup
3. Add it to the playlist

This code is very user-friendly and fun to work with and customize. Choose different values in the Animation Setup section to customize how each animation looks.

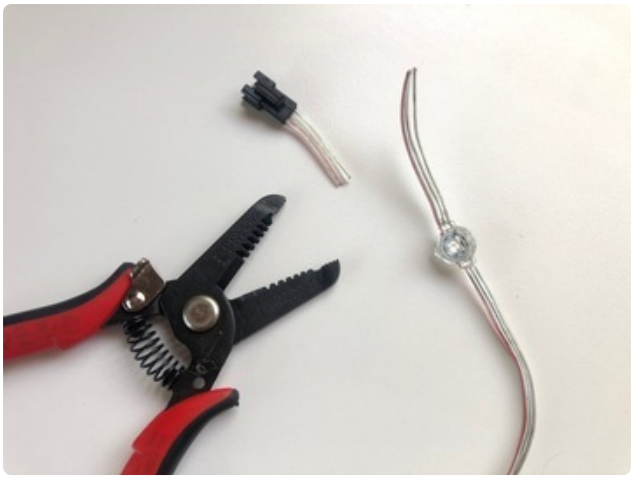
Assembly



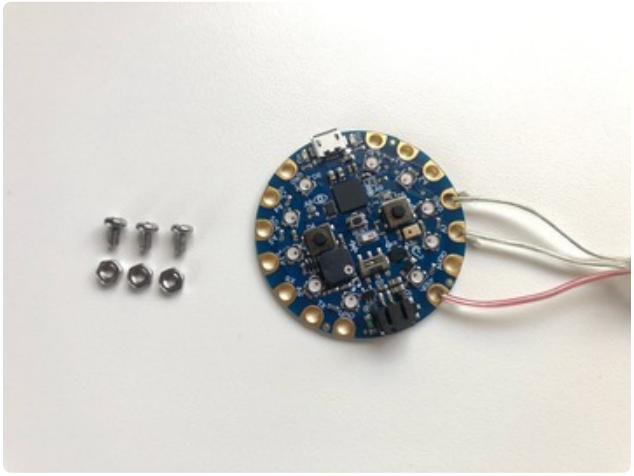
You'll need a screwdriver and some wire cutters / strippers, plus the Circuit Playground Bluefruit, LED strand and bolt-on kit.



Find the IN end of the strip. The lights won't work if you connect to the wrong end. It's written really small on the back so look closely. If you just can't see any writing, align it with the photo and you should be safe. The male connector is usually (but not always) connected to the IN end.

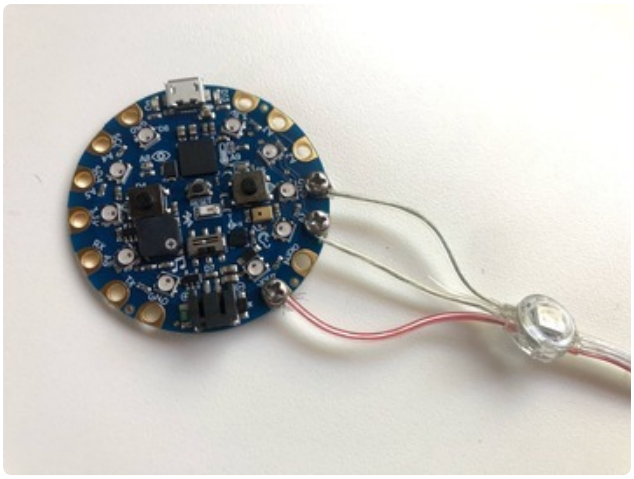


Cut the connector off. Strip about 1/4" of shielding from each of the three wires.

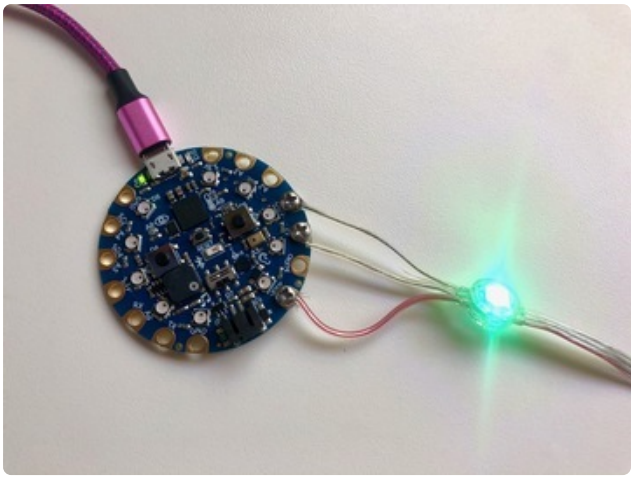


Neaten up the strands with your fingers, then hook the wires through the holes as shown:

Red wire --> VOUT
Middle wire --> A1
Last wire --> GND
The light should be face up with this orientation. If it's face down, you've got the wrong end of the strip! Switch it around and use the other end.



Push a screw through each wired hole and secure it on the back with a bolt. Use a screwdriver to tighten the screw firmly down while holding the bolt tightly against the back of the board with your fingers.



Time to test it! Upload your code, if you haven't already, and make sure the lights come on and that the buttons work to change the modes and variations of the lights.

Troubleshooting

If your lights didn't come on, here are a few things to try:

1. Flip the tiny slide switch on the face of the Circuit Playground the other way. Our sample code uses this as an on/off switch, so make sure it's not just in the "off" position
2. Check your wiring. Is the red wire on VOUT, the middle wire on A1 (not Audio) and the 3rd wire on GND?
3. Try connecting to the other end of the strip -- you may have the "out" end instead of the "in" end.
4. Make sure your wires are neat and tidy, and not touching any of the other pads or causing a short
5. Re-upload your code to make sure it's on there

If nothing helps, head over to the Circuit Playground Bluefruit Guide (https://adafru.it/LC5) for more things to try.

Make the Crystals

Download the Crystal Shapes

I've included two basic shapes, called "Khyber" (on the left) and "Infinity" (on the right). There is a third design (called "Thra") available for download from my Etsy store (https://adafru.it/OMF) (along with pre-cut crystal kits if you're in a hurry!). There are three sizes for each shape, a small, medium, and large. Each design has two pieces that connect together with locking tabs.

Download Crystal Gem Shapes

https://adafru.it/LC6

If you don't have a vinyl cutter you can still do this project. Just print the shapes out and use them as a template. The dotted lines are score lines and the solid lines are cut lines. Feel free to resize them -- just be sure the NeoPixel wires will fit through the rectangular wire holes.

Laminate Your Cellophane

Unroll your cellophane wrap and cut pieces that are the same size as or slightly smaller than your laminating pouches.

I use a rotary cutter (https://adafru.it/19AI) for this and find it goes quickly and gives me nice clean straight edges.

I'm using 3 mil thickness laminating pouches. The 5 mil will also work, but the extra stiffness makes the cutting and folding a lot more difficult, so stick with 3 mil if you can.

Crumple up the cellophane and then flatten it out again to give your material some fun texture. Place the cellophane inside the laminating pouch so no edges are poking out.

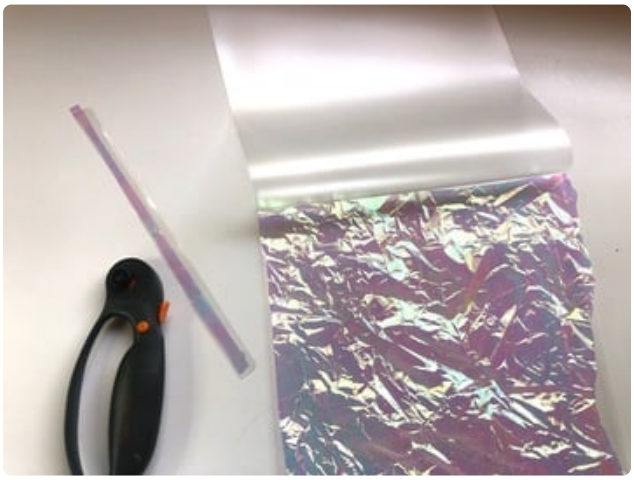
Run it through your laminating machine. I find it helpful to smooth it gently as it comes out of the machine. The flatter and smoother you can get your sheets, the better they'll work with the cutting machine.

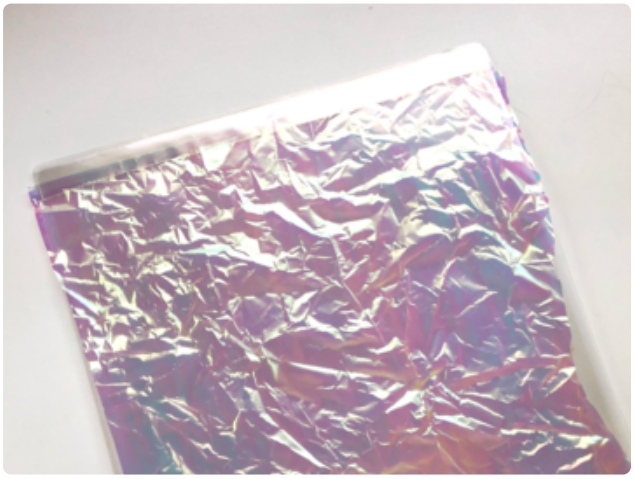
A Note about Laminating Machines

You can find these for very inexpensive prices (around $20) on Amazon or other online retailers. I started out with a slightly pricier Scotch brand machine, then replaced it with a less expensive Amazon brand one when it finally gave up the ghost (after faithfully laminating around 600 sheets, bless its heart).

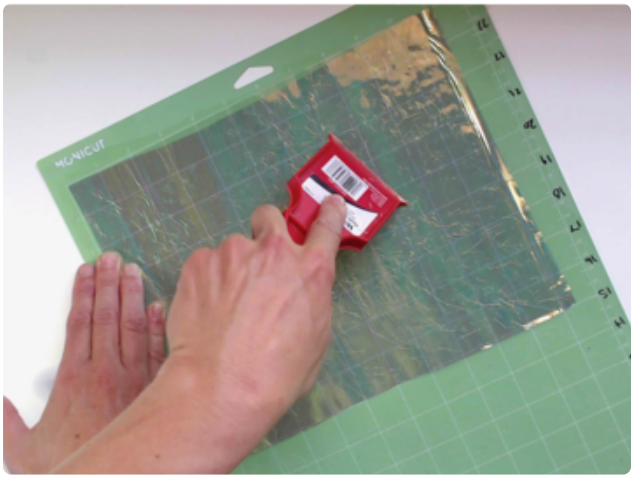
I've had a lot of trouble with the new cheap machine jamming and crumpling up my materials. I came up with a fix, which I'll explain below.. but if I had it to do over again, I'd spend the extra $10 and get a good quality machine. Quality tools make for far fewer headaches.

If Your Machine Keeps Jamming



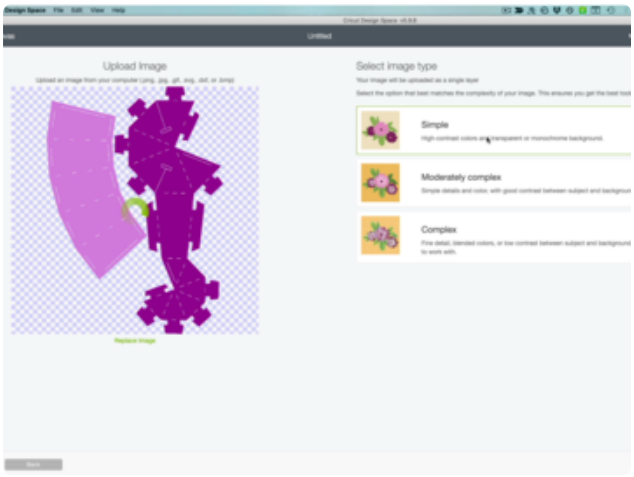


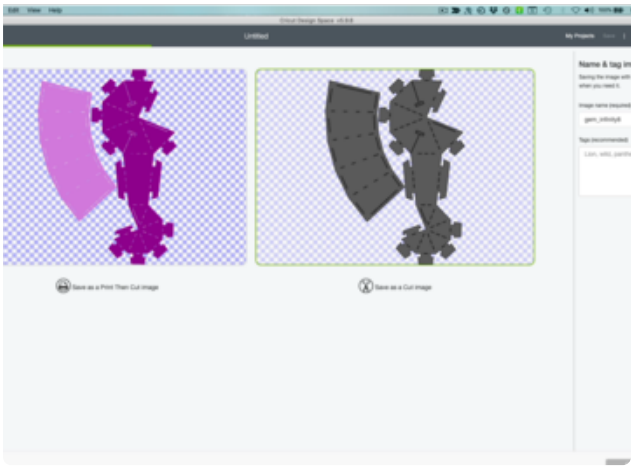
My sheets kept getting stuck and not emerging cleanly from the machine. I fixed this by adding a stiffener across the very top of each sheet. I took one sheet that had laminated successfully, and cut a narrow strip off one edge, then slipped this up inside the unlaminated pouch, against the sealed edge, before running it through the machine. This kept the top edge nice and stiff and really minimized the jamming.



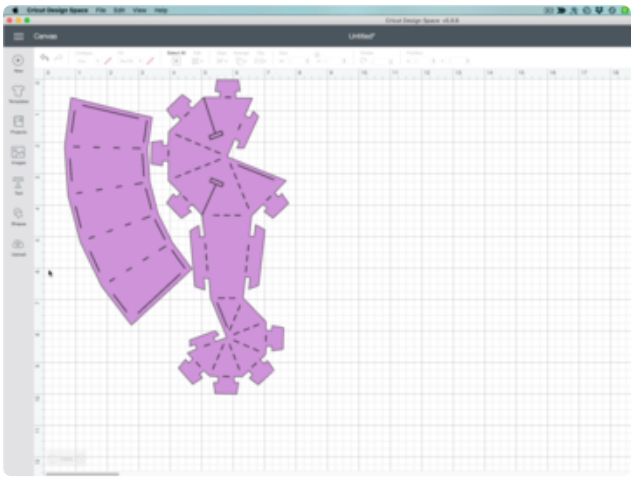
Place your laminated cellophane sheet onto your vinyl cutter's sticky mat and press it down firmly. I use a wallpaper scraper for this -- they're almost free at the hardware store and I haven't found a tool I like better for the purpose. They also work great for cleaning the mats in between use.

Cutting on Cricut Vinyl Cutter





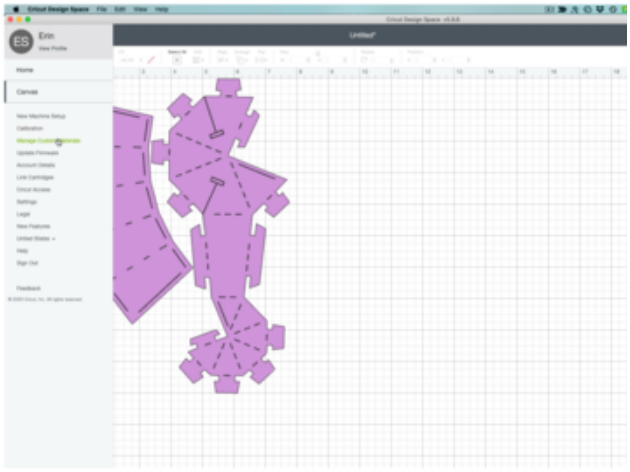
Open the Cricut Design Space app and choose "New Project". Click "Upload" and upload the crystal design of your choice. Choose "Simple" on the next screen, then "Continue" on the next screen since there is no background to remove. On the third screen, select "Save as a Cut Image".



Import the uploaded image into your project. I like to make it a color other than black, so it's easier to see and work with. You'll need to resize it as well -- I've given you high resolution 300dpi images, since the Cricut software does a much better job with those, but that does mean you'll need to size each crystal down.

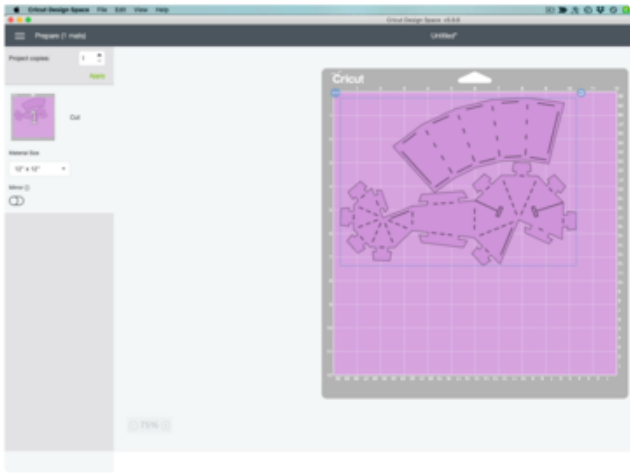
Recommended Sizes

gem_khyber_sm --> 4" high
gem_khyber_med --> 5" high
gem_khyber_lg --> 6" high
gem_infinity_sm --> 4" high
gem_infinity_med --> 6" high
gem_infinity_lg --> 8" high
The Khyber Crystal style is the taller, skinnier style with straight sides, and the Infinity Crystals are wider across with angled sides.



Click the icon in the upper left and select Custom Materials. Scroll until you find "Stencil Film 0.4mm" and select it. Make sure the settings read:

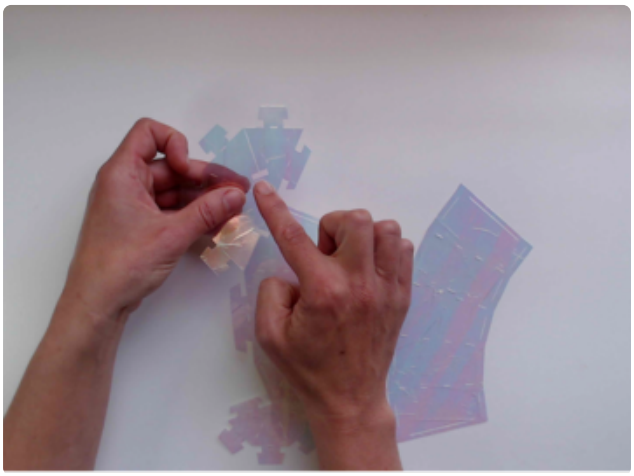
Fine Point Blade
Off (single pass mode)
Pressure should be around 325



Close out of the Custom Materials menu. Once you're happy with your sizing, click "Make It". You have an opportunity to change your layout here, so make sure your crystal will fit nicely on your material.

Set your Cricut dial to "Custom" and select the Stencil Film 0.4mm. Load your sticky mat into the machine and press Go.

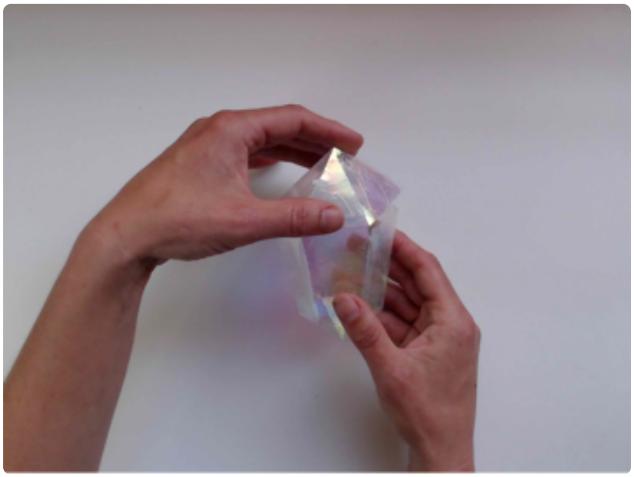
Each crystal cuts in two pieces, with a rectangular hole for the NeoPixel strand wires to poke through.

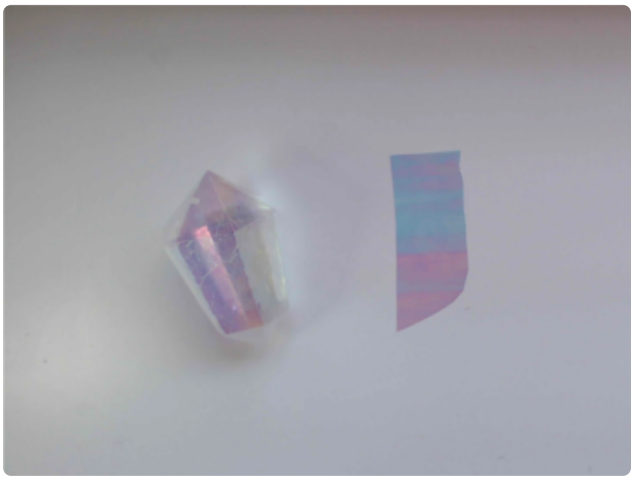


Crease all the fold-lines and tabs. The edges with the NeoPixel wire hole are not scored, because I wanted to make them sturdier, but you'll need to fold those as well. Line up the tabs on either side of the hole segments and crease to the point of the crystal.

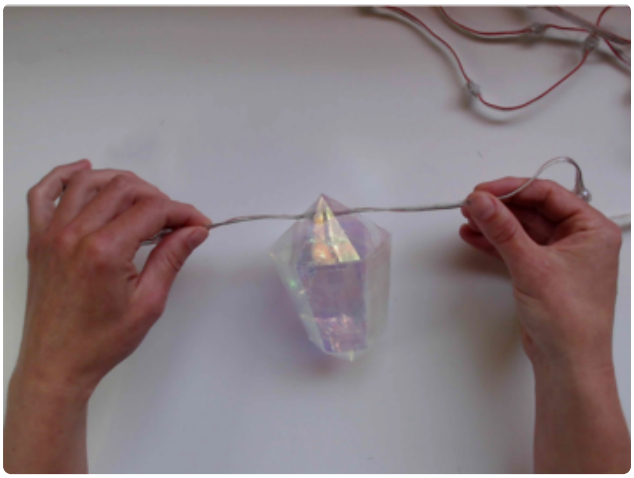
This is more easily shown in the video at the beginning of this guide, so take another look if you can't figure it out.

Insert the tabs from the outside in, so they end up on the interior of the crystal.





Crease all the folds on both pieces, then assemble, leaving the three tabs between the two light holes open for now. Cut and crumple a small piece of un-laminated cellophane and tuck it inside to create the illusory "occlusions" that catch the light and make this crystal so convincing.



Insert the light strand into the two holes, with a light centered inside the peak of the crystal facing downwards. Finish by buttoning up the last three tabs.

If you're just making indoor string lights, the tabs are sufficient to hold them with no glue.

If you're using these for cosplay or other applications where they'll need to stand up to wind or abuse, I recommend adding a dab of glue to a few of the tabs to make sure they stay put. Hot glue or a tiny drop of superglue work great.

Plug in your Circuit Playground Bluefruit and watch your crystals glow!