Infinity Mirror Collar

Created by Debra Ansell

https://learn.adafruit.com/infinity-mirror-collar

Last updated on 2024-03-08 03:55:11 PM EST

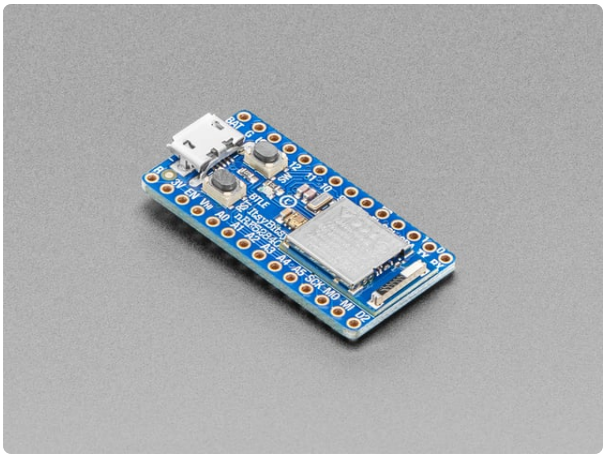
Table of Contents

Overview

An infinity mirror displays endless reproductions of whatever sits between its two reflective surfaces. This wearable design puts a twist on the typical infinity mirror project. By using holographic vinyl as the reflective material and EVA foam as a base, we are able to form the flexible materials into a circular collar, creating an appealing statement necklace.

The two Side Light NeoPixel strips nestled between the vinyl layers produce colorful, dynamic patterns multiplied many times over in the resulting reflections. Controlled with CircuitPython code running on an ItsyBitsy nrf52840 Express, this unique wearable accessory lets you choose custom colors and animation patterns on the go with the Bluefruit phone app interface.

Parts List

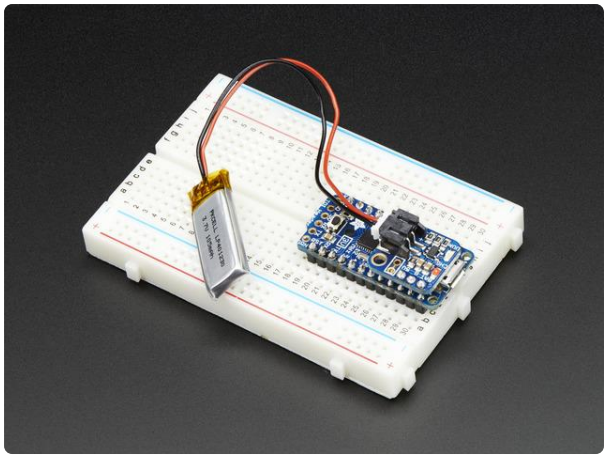


Adafruit ItsyBitsy nRF52840 Express - Bluetooth LE

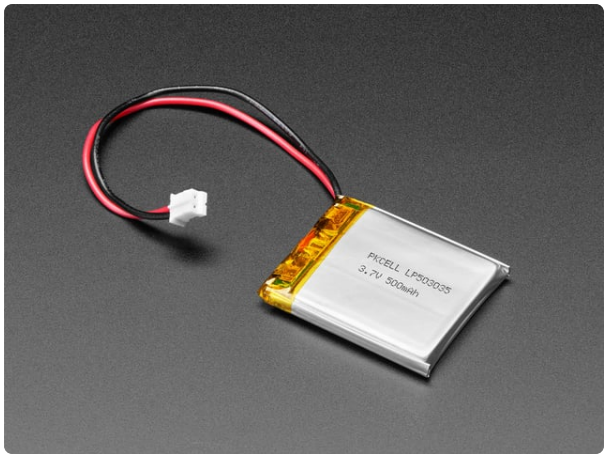
What's smaller than a Feather but larger than a Trinket? It's an Adafruit ItsyBitsy nRF52840 Express featuring the Nordic nRF52840 Bluetooth LE...

https://www.adafruit.com/product/4481

[Adafruit NeoPixel LED Side Light Strip - Black 60 LED]
Fancy new side light LED strips are a great alternative for folks who have loved and used Adafruit LED strips for a few years but want gorgeous, glowy light emitting at...
https://www.adafruit.com/product/3636



[Adafruit LiIon/LiPoly Backpack Add-On for Pro Trinket/ItsyBitsy]
If you have an ItsyBitsy or Pro Trinket you probably know it's the perfect little size for a portable project. This LiPoly backpack makes it really easy to do! Instead of wiring 2...
https://www.adafruit.com/product/2124



[Lithium Ion Polymer Battery - 3.7v 500mAh]
Lithium-ion polymer (also known as 'lipo' or 'lipoly') batteries are thin, light, and powerful. The output ranges from 4.2V when completely charged to 3.7V. This...
https://www.adafruit.com/product/1578

[Breadboard-friendly SPDT Slide Switch]
These nice switches are perfect for use with breadboard and perfboard projects. They have 0.1" spacing and snap in nicely into a solderless breadboard. They're easy to switch...
https://www.adafruit.com/product/805

Additional Parts/Materials:

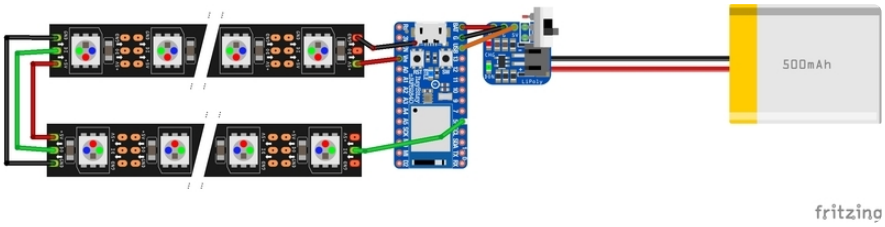
- Transparent holographic vinyl, like this roll from Amazon (https://adafru.it/Qf8) that is partially reflective and approximately 4 mil (0.4 mm) thick. Not all holographic vinyl is transparent so check the description carefully before purchasing.
- One sheet of black, 2mm thick EVA foam with adhesive backing that is 12" wide or wider. Etsy vendors such as this one (https://adafru.it/Qf9) can be a good source for EVA foam sheets.
- Double-sided clear adhesive tape that either comes in a 1/8" roll or in a sheet that can be cut to size. This clear craft tape (https://adafru.it/Qfa) works well
- A 6" length of black, 1/2" wide self-adhesive Velcro (hook and loop) strip, like this one (https://adafru.it/Qfb)
- Painters' tape or other low-tack, easily removable masking tape
- Electrical tape
- A sturdy cylindrical base, such as large aluminum can, to build the collar around
- Optional but helpful: a good glue for sticking EVA foam to itself, such as Gorilla Clear Glue (https://adafru.it/Qfc)
- Hookup wire
- Shrink tube

Tools:

- Soldering iron/solder
- Wire cutter/stripper
- Scissors
- Pen
- Printer and printer paper
- Craft knife
- Cutting mat or surface
- Ruler/straightedge

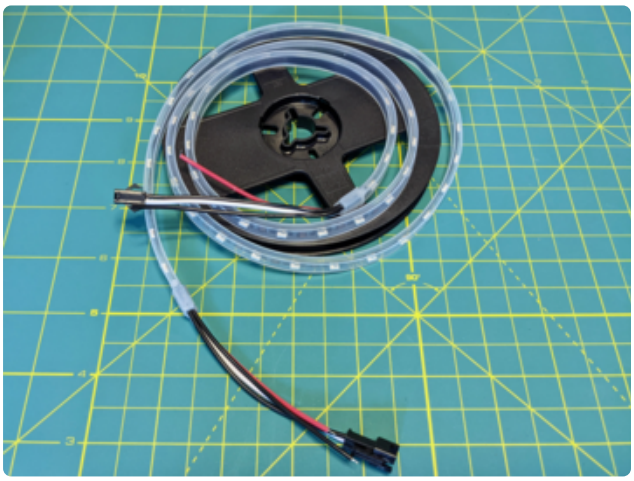
Assemble the Electronics

Circuit Overview



The circuit diagram above shows this project's electronics and wired connections. The two Side Light NeoPixel strips are controlled by an ItsyBitsy nRF52840. Power is provided by a 500 mAh LiPo battery via a LiPoly Backpack which also recharges the battery when power is connected to the Itsy Bitsy board through its USB connector. We'll solder a slide switch to the backpack to make it easy to turn the battery power on and off.

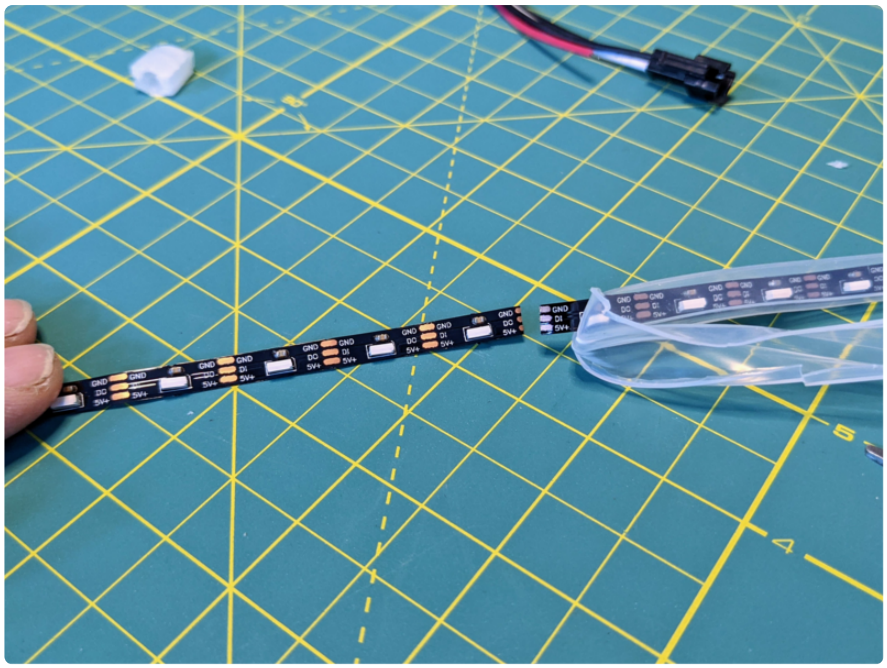
Cut the Side Light NeoPixel Strips



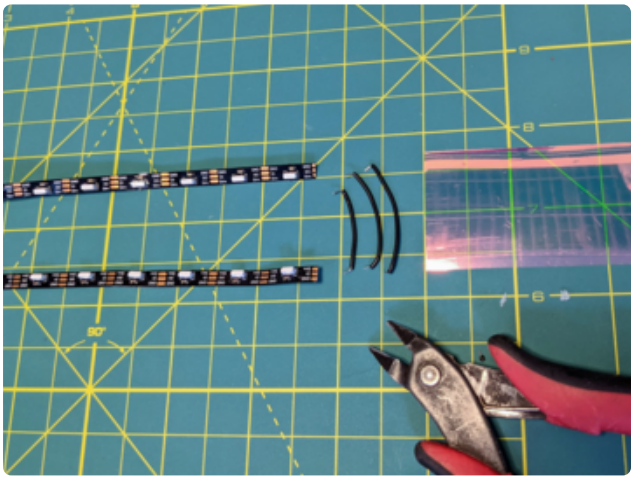
This project uses two 12-pixel lengths of NeoPixel Side Light LED strip. Unlike standard NeoPixel strips, the LEDs on the Side Light strip emit light perpendicular to the strip's surface, allowing a viewer to see the illumination from the LED without having to look directly into a very bright light source.

Identify the segments of strip to cut. You'll need to remove the protective silicone casing around them. The Side Light NeoPixel Strip has pre-soldered power/signal connectors at both ends, but these wires are too bulky to be included in the collar and should not be included in the cut pieces.

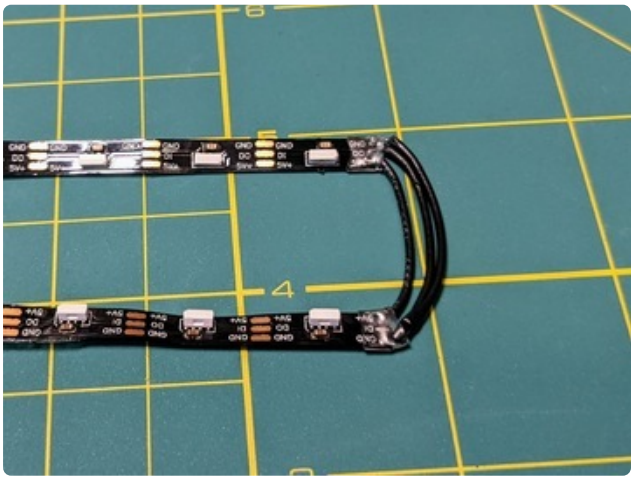
When cutting the strip, make the cuts directly across the copper pads - as close to the midline as possible, so that there will be partial pads to solder on both ends.



Solder the Side Light Strips



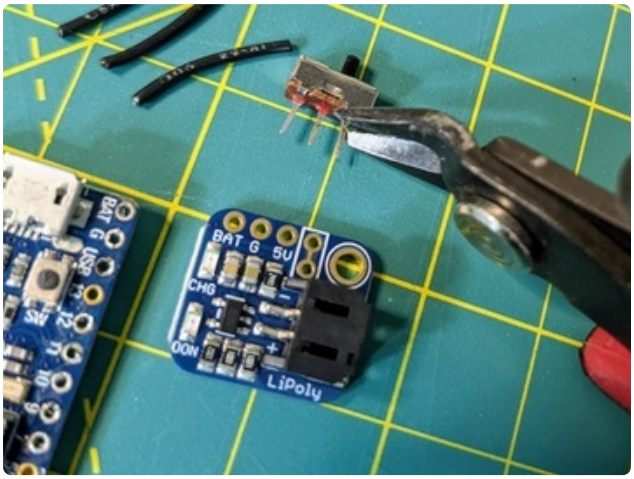
Solder the output pads of one LED strip to the input of the other. The LED strips sit parallel to each other with outer edges spaced 36mm apart. Place the strips on a flat work surface and align their edges with the largest piece of vinyl as shown, making sure the LEDs on each strip point towards each other. Cut three pieces of hookup wire in the correct lengths to connect the pads.

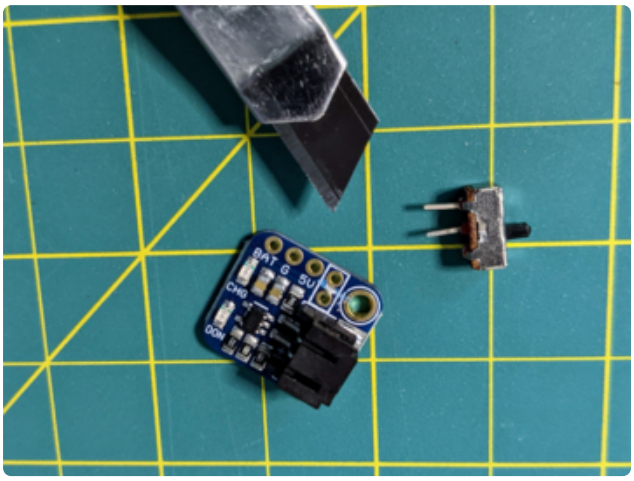


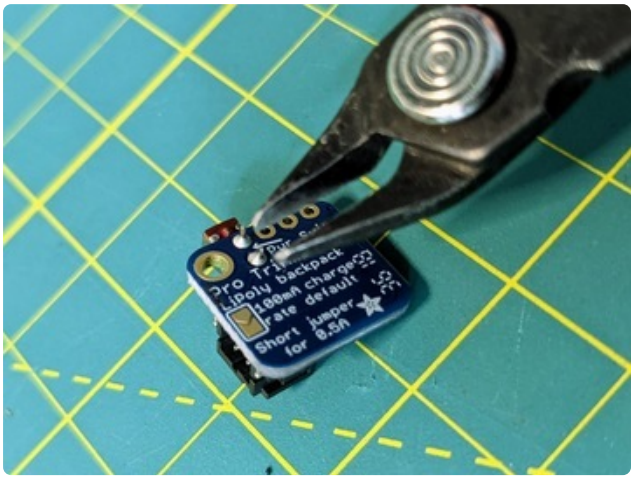
Read this tutorial (https://adafru.it/M9F) for a good overview of LED strip solder techniques. First tin the pads you'll be connecting. Then strip the ends of the wires and tin them with a bit of solder. Once both have been tinned, place the wire end over a pad, and apply a bit of heat with the soldering iron tip until they melt together.

Repeat this step for all three wires, connecting 5V to 5V, GND to GND and Dout to Din. After soldering, slide a small length of shrink tube over each LED strip to cover the solder joints, then heat with a heat gun or hair dryer to shrink the tube and provide strain relief.

Solder the Slide Switch



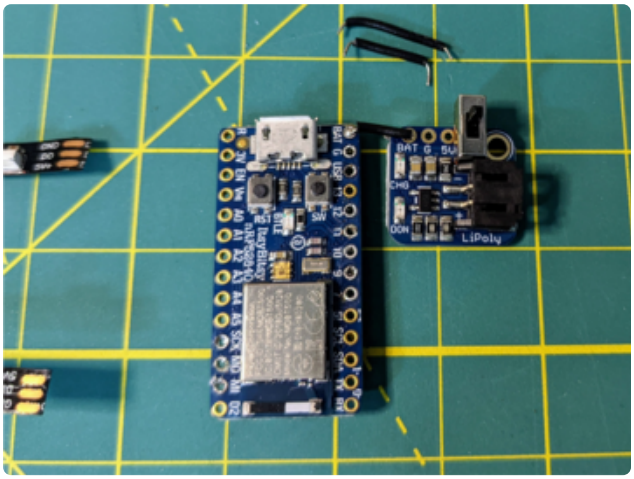




The LiPoly backpack board has pins designed for attaching a slide switch to control its power output. Before soldering the switch you'll need to sever the trace connecting those pins with a craft knife. We'll only need two of the switch legs, so using tin snips, clip one of the outer legs off of the switch.

Slide the two switch legs through the holes in the LiPoly backpack and solder them in place. Once soldered, you can trim the portions of the switch legs which protrude past the bottom of the board using a pair of snips.

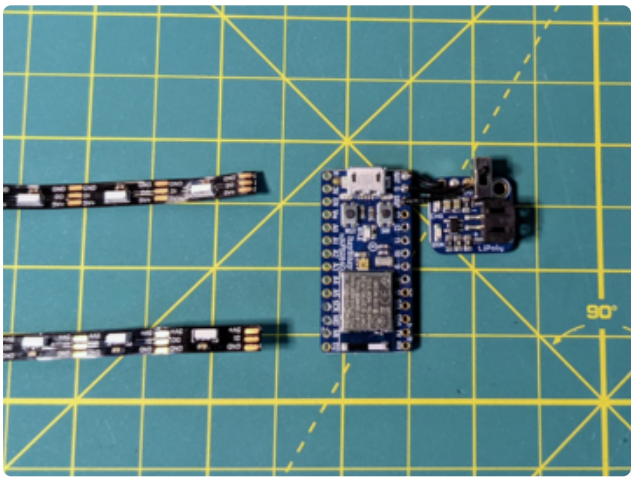
Wire the Backpack



Connect the corresponding Bat, GND and USB pins on the LiPoly backpack and the ItsyBitsy nRF52840 with short lengths of hookup wire as shown. While the LiPoly backpack can be mounted directly on top of the ItsyBitsy with header pins, we want them to sit side-by side in the collar to create lower profile. When joined, the boards should sit approximately 5mm apart.

Its easiest to complete this step by first soldering the Bat wire to both boards. That helps to hold them in position while you solder the GND and USB wire in place.

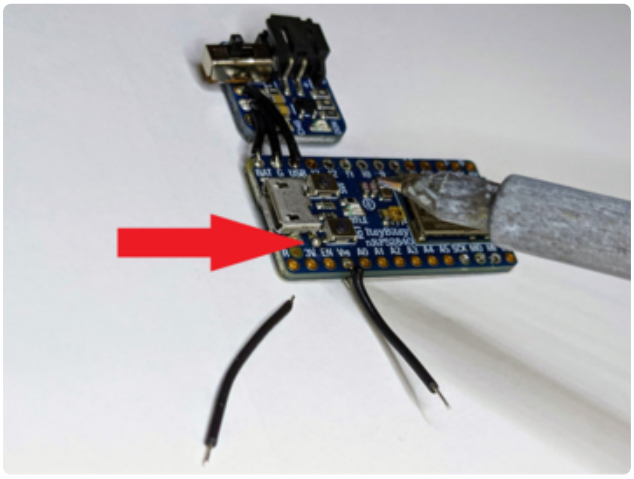
Connect the ItsyBitsy to the Side Light Strips

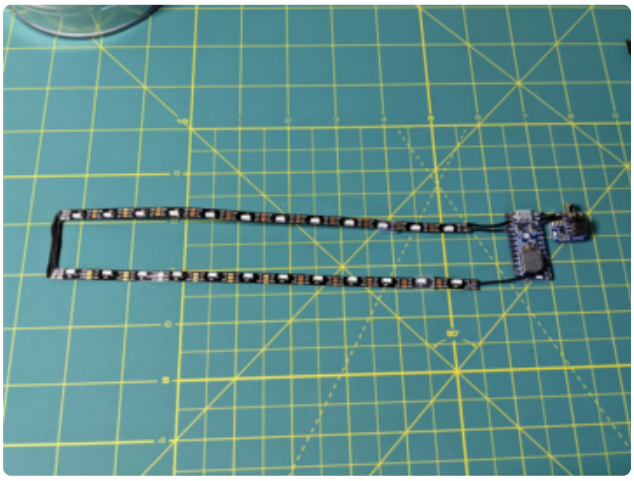


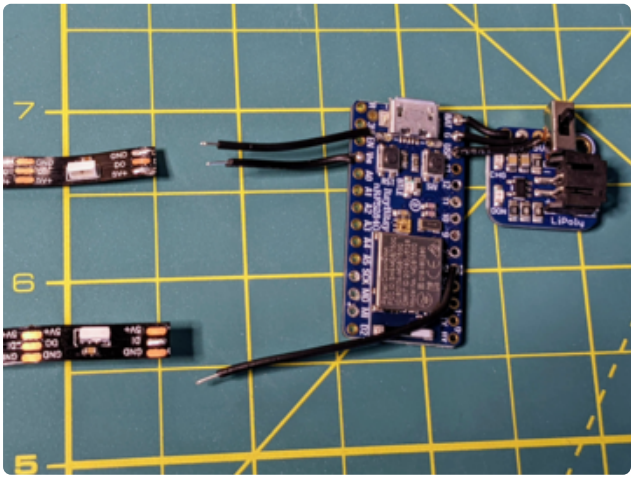
Next we'll solder the 5V, GND and Di pads on the strips to the appropriate board pins, leaving approximately 3/4" of space between the ends of the LED strips and the edge of the Itsy Bitsy nRF52840 board..

The LED strip 5V pad connects to the Itsy Bitsy's Vhi pin. From the ItsyBitsy nRF25840 description, we see that the Vhi will always carry the higher of the USB and Battery voltage when they are connected, allowing the NeoPixel strip to run off of either battery or USB power.

In the electronics assembly the Vhi pin is located close to the output end of the upper LED strip, so it will be easiest to connect it to the NeoPixel 5V pad there. Since power and ground on both LED strips are connected to each other, it doesn't matter which strip segment we use to make those connections. Solder a piece of hookup wire to the ItsyBitsy's Vhi pin, but don't solder it to the LED strip just yet.







Next we'll add ground wire. Note that we've already connected the ItsyBitsy's GND pin to the LiPoly backpack, and there isn't another ground pin on the board. We can use a little trick to create a ground connection between the ItsyBitsy nRF52840 and the Side Light NeoPixel Strips without an additional pin. Since ItsyBitsy's microUSB connector is attached to ground through its solder pads, we can solder a ground wire where the microUSB connector meets the board.

Carefully use your soldering iron to place a small blob of solder onto one one of the pads holding the USB connector (shown in the image with the red arrow). Then tin the end of the ground wire and use the tip of the soldering iron to melt the wire to the pad.

Now solder a slightly longer wire to pin 5 on the ItsyBitsy. This wire will connect to the Di pin on the bottom LED strip. Lay the ItsyBitsy next to the LED strip ends to make sure the wires extend the same distance away from the board. Tin the corresponding pads on the LED strips.

Before soldering the wires to the NeoPixel strips, cut two small pieces of shrink tube and slide them over the LED strip ends. Now solder the 5V, GND and Di wires to their corresponding pads, then slide the shrink tube over the connection and apply heat to shrink it. The final electronics assembly, minus the battery, should look like the last picture at left.

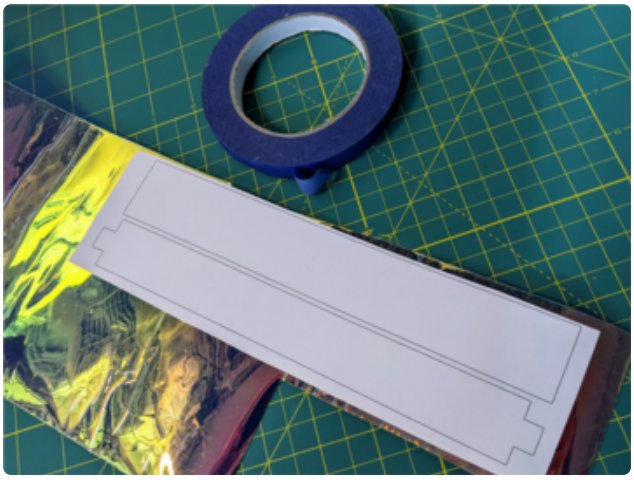
Prepare Craft Materials

Cut the Holographic Vinyl

NEVER USE A LASER CUTTER TO CUT VINYL, AS IT WILL RELEASE TOXIC CHLORINE GAS

HolographicVinyl.svg

https://adafru.it/Qfe



Print the the holographic vinyl template from the SVG file provided by the link above. Once it has printed, verify that it is the correct size by checking the 1" reference square against a ruler.

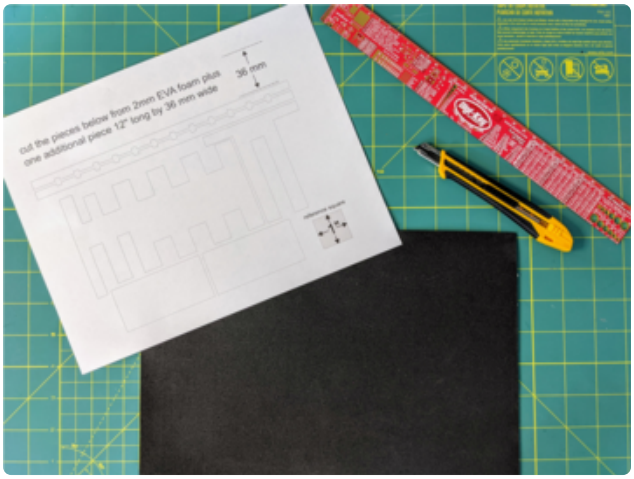
Next, cut away the extra paper around the printed pieces, leaving a small margin around the outlines of the two pieces. Unroll and smooth out the holographic vinyl, then place the template on top with the printed side facing the vinyl. Using tape (painter's tape is fine), secure the template to the vinyl by placing tape around all the edges as shown.

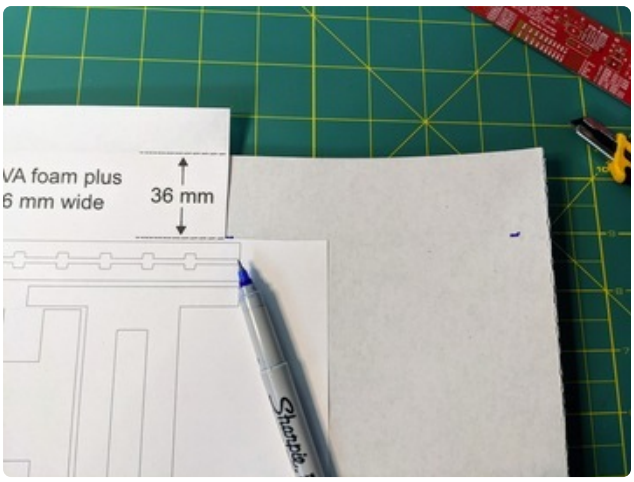
Turn the vinyl and template over. You will be able to see the outlines of the two pieces through the translucent vinyl. Using sharp scissors, cut around the two piece outlines, starting at the middle of the template and cutting the taped edges last.

Cut the EVA Foam

EVAFoam.svg

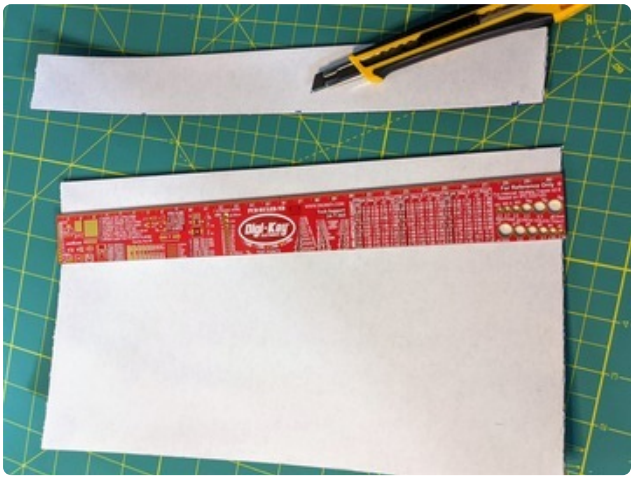
https://adafru.it/Qff

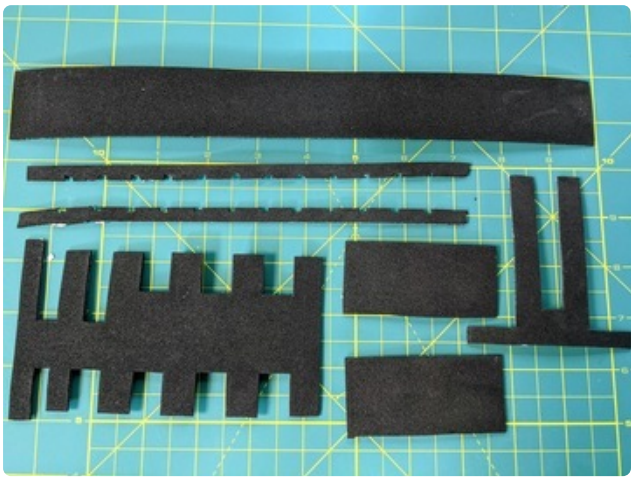




If you have a laser cutter or vinyl cutter that can handle 2mm thick EVA foam, you can import the vector template for the EVA foam pieces directly into your software and cut them using your machine. You will need to cut an additional rectangular piece that is 12" long by 36mm high. If you don't have a machine to cut these pieces, it is straightforward to cut out the pieces by hand using the method described below.

Print the template for the EVA foam from the SVG file provided, and check the scale using the 1" reference square included on the page. This template contains outlines of every foam piece except the largest one, which is too long to fit on printer paper.





The largest foam piece is a rectangle 12" long by 36mm wide. While the template doesn't have an outline of this piece, it does contain two lines separated by exactly 36 mm which you can use to draw cutting lines directly on the foam. Flip the EVA foam over so the paper backing faces upwards, and, using the markings on the template, create a series of marks spaced 36mm from the edge of the foam sheet. The EVA foam sheet shown here is exactly 12" long, so it was straightforward to line up a ruler with the marks and cut along the edge of the ruler with a craft knife. Alternatively, you can draw a straight line joining the markers with a pen and ruler, then use scissors to cut out the rectangle.

Once the large EVA foam piece is cut, trim the template around the group of remaining pieces. Using a glue stick or double-sided craft tape, stick the template to the paper on the back of the foam. You can then cut along the printed outlines with scissors or a craft knife to create the rest of the foam shapes.

The small notches in the edge of narrowest foam pieces can be tricky to follow with a regular scissors. To cut the notched piece accurately, It's easiest to cut a straight line directly along the notched edge first, ignoring the notches, and then go back to cut them out afterwards. The small details are most easily created using a craft knife aligned with a ruler or a small, sharp scissors, such as embroidery scissors.

When finished, you should have all the pieces shown in the final image.

Find a Build Model

The components of the collar fit together in curved layers. The different materials in the build have varying degrees of flexibility and stretch. In order to prevent the collar from wrinkling or pulling when worn, it must be constructed with the correct curvature.

You'll need to find an appropriately sized "model" of your neck to build it around. Measure the circumference of your neck using a piece of string or cloth measuring tape, and search your surroundings for a sturdy cylindrical object with a similar circumference. The closer you can get to your actual neck size, the better. I found a 28-oz can of tomatoes that was within 1/2 inch of my neck's circumference, and removed the label to make it easier to work with.

Construct the Collar

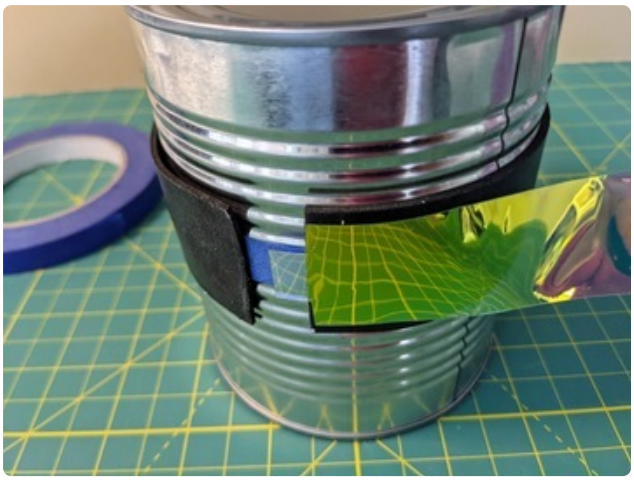
Place the Foam Base

The large 12" x 36mm EVA foam rectangle forms the base of our collar. To start, we'll wrap it around the cylindrical "model" to give it the correct curvature.

Take a 3-4" piece of painters' or other low-tack tape and stick it onto the non-adhesive face of the EVA foam rectangle so that about half the tape extends past the edge of the foam. Then wrap the rectangle around the can with the sticky face of the painters' tape facing outwards. Push the other end of the rectangle firmly onto the free end of the tape so that the EVA foam is secured around your model.

Once the EVA foam is securely fastened in place and won't slide off, peel the protective paper away from the adhesive side of the foam.

Attach the Inner Vinyl Layer

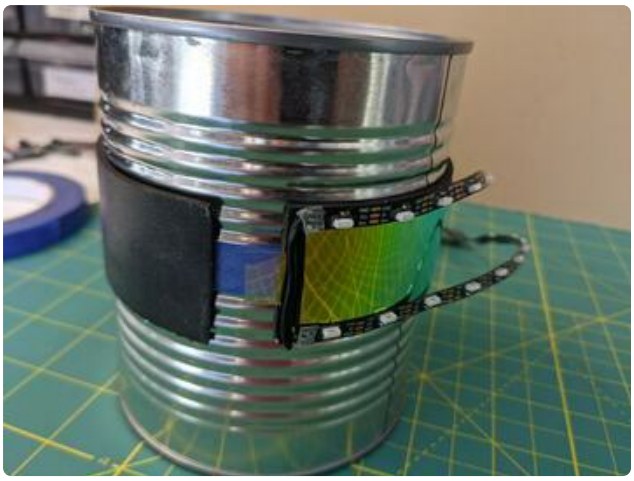


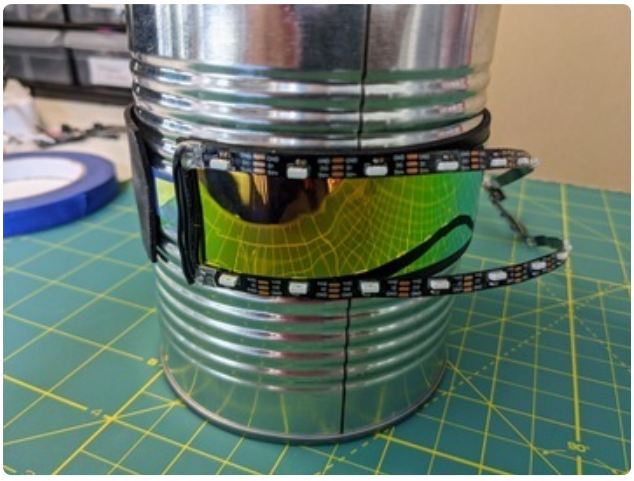
The inner layer of holographic vinyl has tabs protruding from both its ends. Take this vinyl piece and peel off any protective paper that may be covering it.

Carefully - taking the time to get the alignment right - stick the vinyl to the EVA foam base. The tab at one end of the vinyl should overhang the foam completely so that the base of the tab, where the vinyl becomes wider, aligns with the end of the foam base. Be sure to center the vinyl vertically relative to the foam. The vinyl is narrower than the foam, leaving a margin of about 4-5 mm of EVA foam extending above and below the vinyl.

Wrap the vinyl completely around the EVA foam - keeping it as centered as possible. You'll want to perform this step slowly and carefully. The sticky face of the foam will hold the vinyl in place securely, but also makes it tricky to re-position once placed. It is helpful, when applying the vinyl over the foam, to hold the vinyl tautly out from the can as you wrap it around the foam piece. The vinyl is shorter than the foam and will not reach the end of the foam base when fully adhered.

Align the Electronics





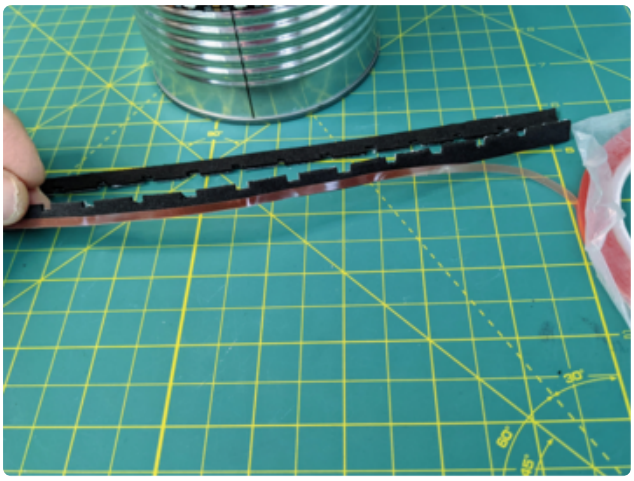
Take the wired electronics without the LiPo battery, and place them near the base of the collar. Starting at the point where the two Side Light NeoPixel strips connect to each other, we'll wrap the LED strips around the foam so that they overlay the sticky exposed portions of the EVA foam extending just above and below the vinyl.

Align the electronics assembly so that hookup wires wires sit at the very end of the foam where the vinyl tab meets the foam base. Carefully bend the LED strips around the collar base, gently pressing them into the sticky foam. The outer edges of the NeoPixel strips should sit flush with or just a bit inside the outer edges of the foam.

When you reach the ItsyBitsy nRF52840 and the LiPoly backpack, press them into the sticky foam as well. The EVA foam adhesive should be strong enough to hold them in place without additional support.

Gently go back over the full lengths of the Side Light NeoPixel strips, pushing them into the foam with your fingertips, while smoothing them out to make sure that there are no wrinkles or bumps between the foam and the strips.

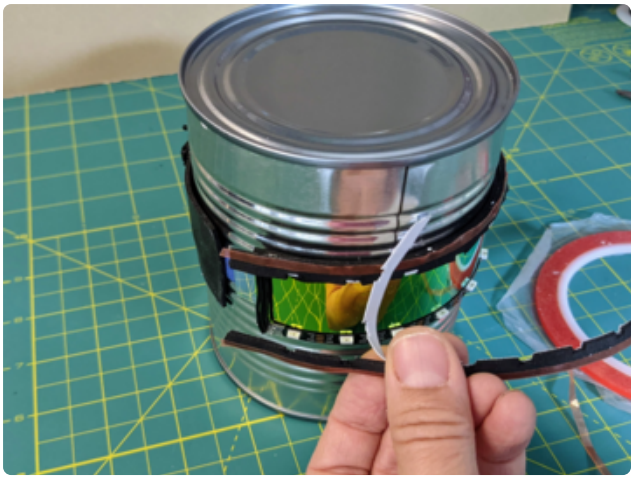
Stick the Foam Spacers

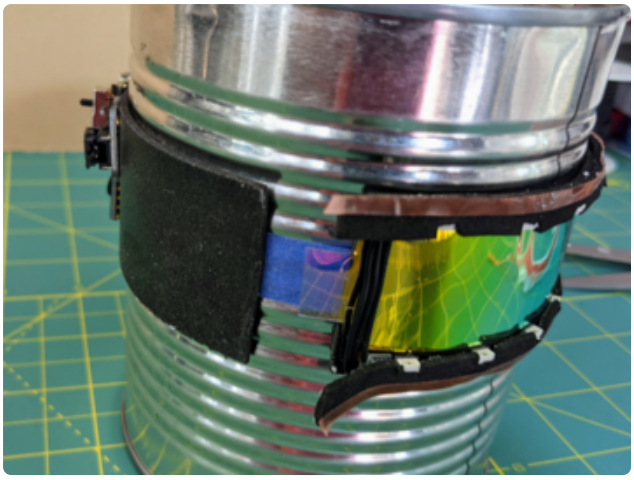


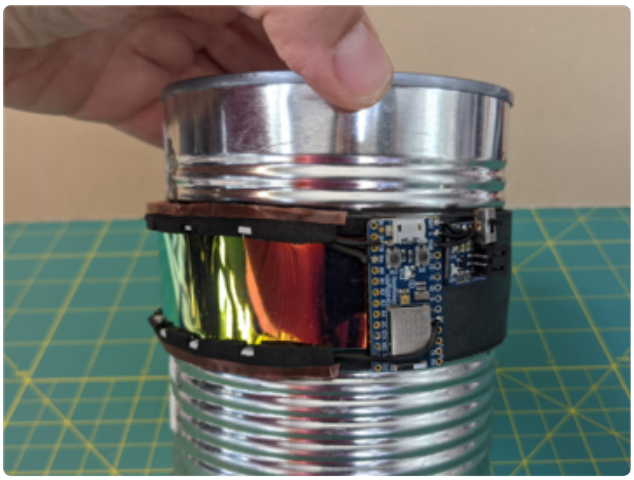
Locate the two skinny notched EVA foam strips. These pieces cover the LED strips and sit between the vinyl layers to keep them spaced a uniform distance apart. The EVA foam's adhesive backing will stick it to the inner vinyl layer, and we'll add a strip of 1/8" clear double-sided tape to the non-sticky side of each spacer so it can adhere to the outer vinyl layer as well.

It's easiest to attach the double-sided tape to the foam before the foam is attached to the collar. Peel a length of the 1/8" double sided tape and press the sticky side into the foam spacers. Run the tape along the entire straight (non-notched) edge of the foam. Leave the backing on the double-sided tape for now.

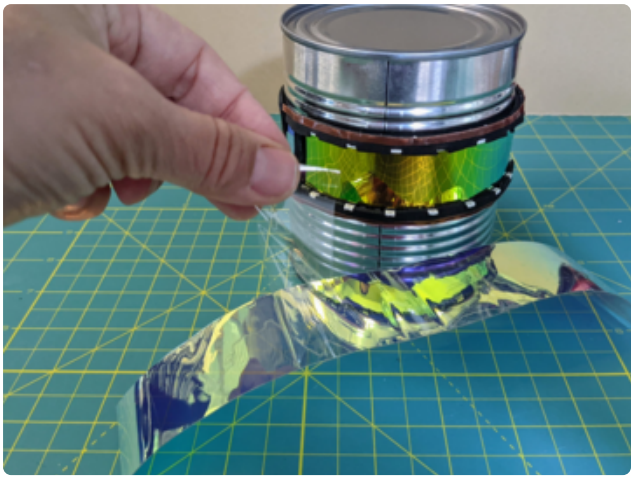
Once you've applied the double-sided tape to each EVA foam spacer, peel the paper backing from the EVA foam and carefully stick one spacer over each of the Side Light NeoPixel strips. Align the foam spacers with the Side Light NeoPixel strips so that each of the 12 pixels fits snugly in its own foam notch, as shown. When in position the ends of the spacer strips will overhang the foam base.

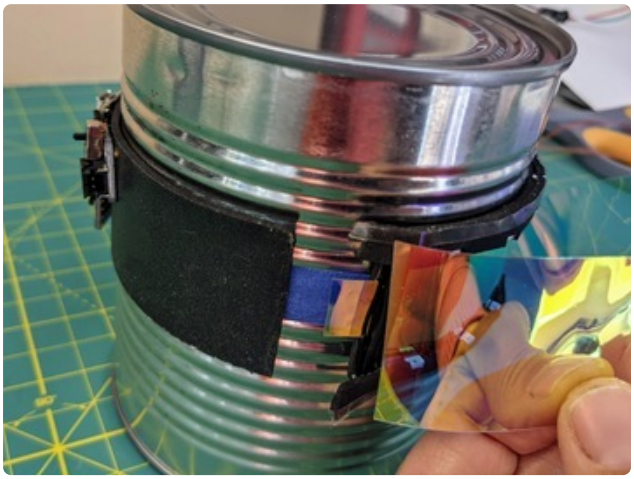






Attach the Outer Vinyl Cover

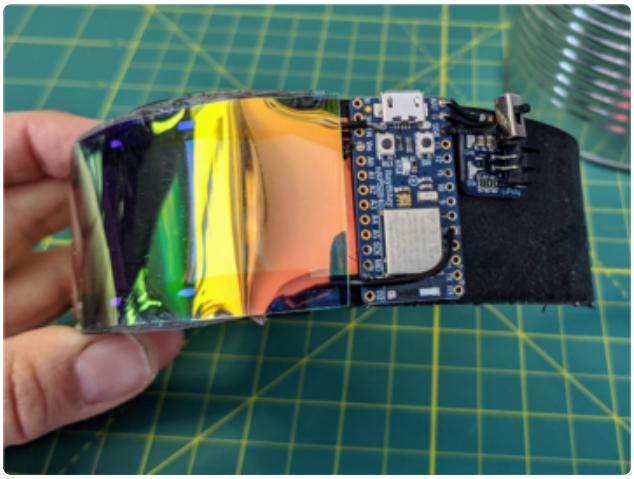


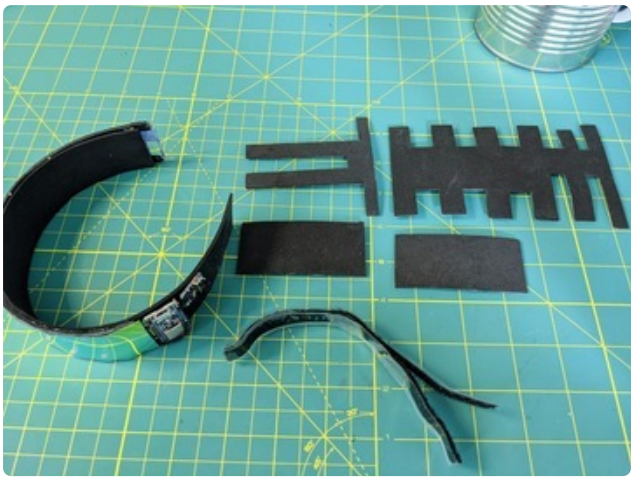


Take the last piece of holographic vinyl and peel off any protective paper it may have. Peel the backing from the double-sided tape on both of the foam spacers.

Carefully align and stick the vinyl to the spacers, starting at the side of the collar without the ItsyBitsy board. The vertical edge of the vinyl should line up closely with the edge of the foam base piece. Start by attaching the vinyl at this edge, then traverse the length of the collar pressing the holographic vinyl into the double-sided tape with your fingertips, smoothing it out as you go to avoid wrinkles.

When in place, the other end of the vinyl should extend approximately to the edge of the ItsyBitsy, as shown. If you find, after smoothing everything out, that part of the foam spacers extend past the edge of the vinyl at the top or bottom edge of the collar, you can carefully trim the foam overhang away with a sharp scissors or craft knife - being very careful not to cut into the LED strips.





The main body of the collar is now complete and you may remove it from the cylindrical "model" for the next steps. The collar should remain curved now, because of the way its layers are stuck together.

Gather the remaining pieces of EVA foam, shown in the first picture at left, as well as the 6" segments of Velcro. They'll be used to extend the length of the collar and create an adjustable fastener that will hold the collar around your neck.

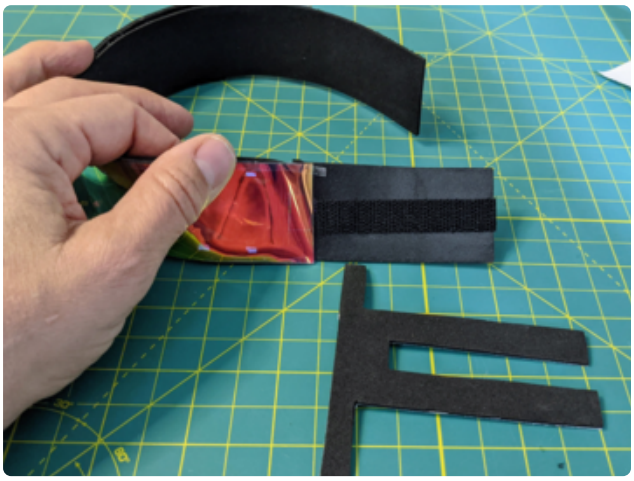
Take the hook side of the Velcro (the scratchier and less soft of the two pieces) and one of the small EVA foam rectangles. Peel back the adhesive backing from one side of the foam rectangle, and attach the exposed sticky side of the foam rectangle to the end of the collar where the holographic vinyl tab juts out. When adhering the foam piece to the vinyl tab, be sure that the sticky face of the foam faces the outside of the collar, and that you align the edge of the foam rectangle with the edge of the collar's foam base as shown.

After the foam rectangle is attached to the collar, remove the rest of its paper backing. Next, remove the backing from the hook side of the Velcro to expose its sticky back side. Then, carefully position the hook side of the Velcro so that its end just meets the edge of the holographic vinyl and it is positioned in the middle of the sticky foam rectangle. Once aligned, press the Velcro into the EVA foam so that they stick together. Trim off any Velcro which extends past the very end of the sticky foam piece.


90°

Add the First Velcro Connector

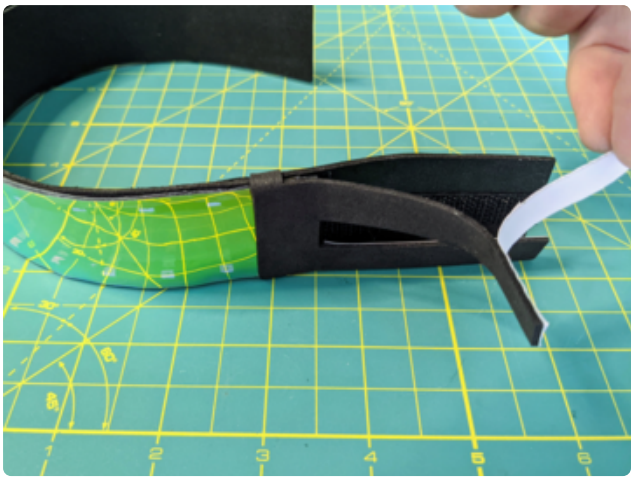
Secure and Cover the Velcro



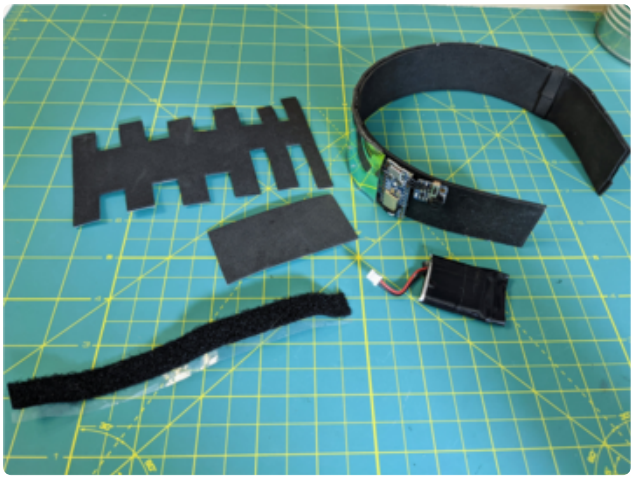
Now we'll use EVA foam piece shaped like the Greek **π** symbol to reinforce the connection between the collar body and Velcro strip extension. Peel the paper backing off the top and "arms" of the **π**-shaped piece as shown, leaving the paper on its "legs" so it will be a bit easier to handle.

Place the foam piece on the front of the collar so that its widest edge overlaps the holographic vinyl by about 5 mm, and it is centered vertically along the collar. Next turn the collar around to view its interior surface. You'll see that the protruding "arms" of the foam piece are inline with the join between the main collar body and the EVA foam piece with the Velcro. Fold the two sticky arms over the top and bottom collar edges so that they cover the join. Press them firmly in place with your fingertips.

Now turn the collar back around so that you are viewing it from the outside. Carefully peel the remaining paper cover from the legs of the Pi-shaped foam piece and stick them to the rectangular foam piece holding the Velcro. The Velcro strip should fit nicely inside the gap between the foam legs.

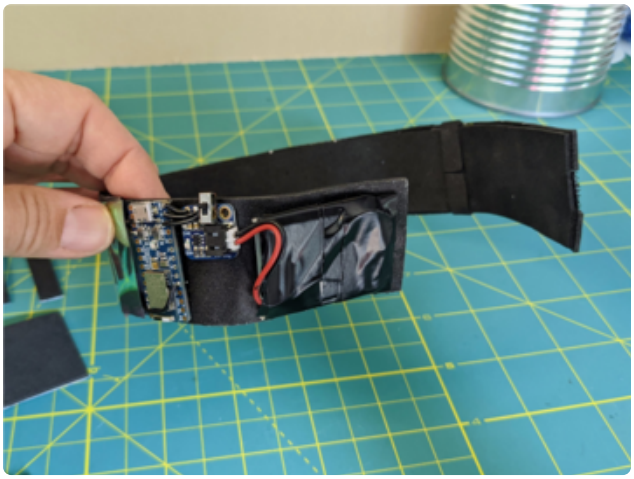


Attach the LiPo Battery

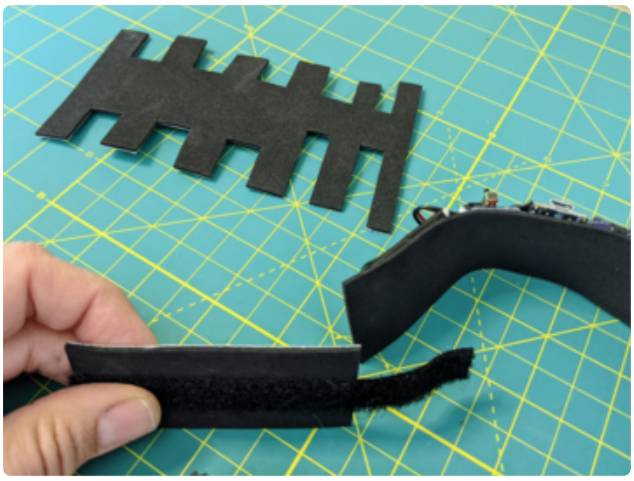


Next, we'll add the battery to the electronics. Since LiPo batteries can be fragile, and our battery will be sitting in sticky EVA foam, we want to avoid damaging it if it needs to be removed or repositioned. Before adding it to the collar we'll cover it with a layer of electrical tape. Cut two or three lengths of electrical tape which are just long enough to wrap around the battery, and wrap the battery in them. When wrapping lay the JST connector wires across the body of the battery and wrap the tape around them, leaving about an inch of the connector wires extending from the tape.

Once it is wrapped, slide the battery connector into the LiPoly backpack and stick the battery to the adhesive EVA foam as shown at left. Orient the battery so that it is centered vertically in the foam and does not extend past the top or bottom edges of the collar.

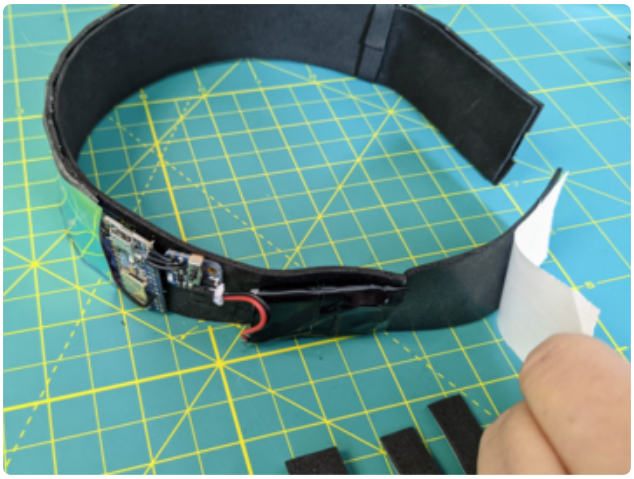


Complete the Velcro Connection

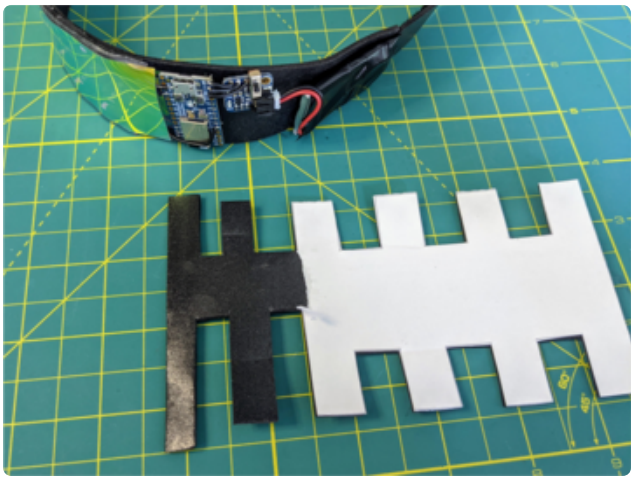


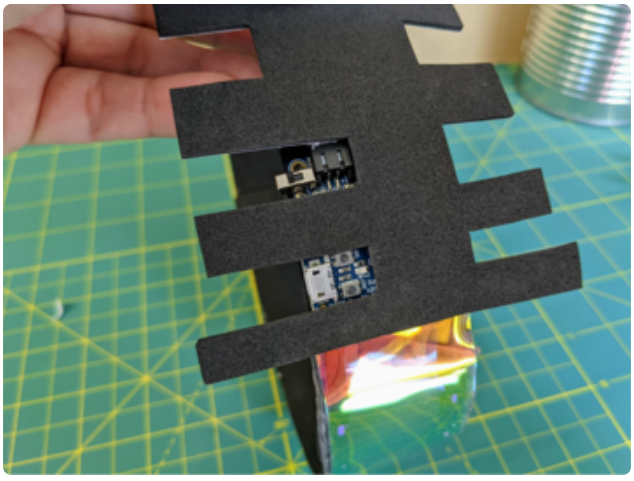
The remaining rectangular foam piece will hold the loop (soft) side of the adhesive Velcro strip to the collar. Leave the paper backing on the EVA foam rectangle to start. Peel the backing from the loop side of the velcro, and stick it to the *non-adhesive* side of the EVA foam. Position the Velcro strip so that one end aligns with a short edge of the foam rectangle and it runs along the center of the foam piece. The loop side of the Velcro strip will overhang the foam by about two inches.

Next, we'll attach the foam rectangle to the side of the collar containing the battery using the sticky back of the overhanging Velcro. Hold the foam rectangle and Velcro strip next to the edge of the collar with the Velcro on the collar's interior. Line up the short edge of the EVA foam rectangle so that it meets but does not overlap the edge of the collar. Once aligned, press the Velcro to the interior of the collar to connect the EVA foam rectangle to the main collar. Once firmly attached, turn the collar so you are viewing its outside surface and peel the paper backing from the foam rectangle you just attached.



Wrap it Up





The final piece of EVA foam wraps around the electronics to help secure and protect them. This foam piece has a number of notches and tabs which will wrap around the collar's top and bottom edges. Two of the notches on one side of this foam piece are a bit deeper than the others. Peel the protective paper off just the side of the foam piece containing the long notches, as shown at left.

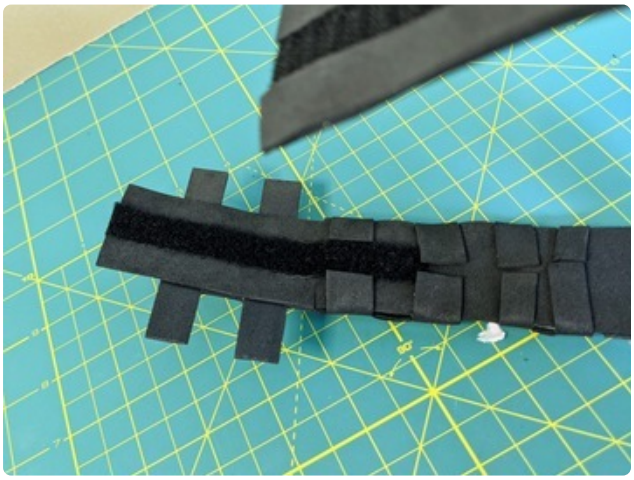
Carefully position the foam piece over the collar so that the first notch is centered around the Itsy Bitsy nRF52840's micro USB connector and sits below the RST and SW pushbuttons. The second long notch is situated around the LiPoly Backpack's slide switch and JST connector as shown. The base of the short notches on both sides of this foam piece should just line up with the top and bottom edges of the collar, and the widest side of the piece should overlap the outer layer of holographic vinyl by a few mm.

When this foam piece is positioned correctly, press the portion of the piece without the paper backing over the electronics to stick it down. Wrap the long tabs around the top and bottom of the collar pressing them down to stick them to the collar's interior.

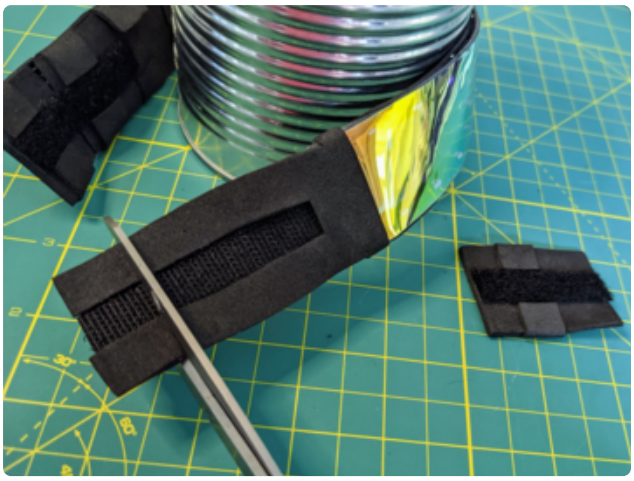
Remove the rest of the paper backing from the foam piece and stick it to the exterior of the collar. The sticky side of this foam piece will stick firmly to the sticky side of the rectangular foam piece holding the velcro.

Turn the collar around so that you can see the interior. One at a time, fold each of the tabs over the top or bottom edge of the

collar and push them to the interior side of the collar to stick them down. The tabs should extend just to the edge of the Velcro. If they overlap a bit, cut them back with scissors so that no foam is covering the Velcro strip.



Trim It Down



The collar structure is complete, but, depending on your neck size, will likely be a bit too long to wear comfortably. You may shorten it by trimming the ends with a sharp scissors. Before trimming, test the sizing by wearing the collar around your neck. If it feels too bulky, then place the collar around your cylindrical "model" and a little bit at a time, trim equal length pieces from both ends until it fits your neck comfortably.

Wrap Up Loose Ends

Depending on the strength of the EVA foam's adhesive backing, you may find that some tabs which wrap around the edges of the collar start to lift up. If this happens, you can fasten them more securely with glue that is formulated to hold foam.

There are a number of foam glues on the market. This clear Gorilla Glue (https://adafru.it/Qfc) works well. Simply apply a very small drop of glue to the tab's inner surface, and secure it down with a clip while it dries. It will stay in place permanently after that.

Glow, Baby, Glow!

Now that the collar's construction is complete, you can test it out by connecting the ItsyBitsy nRF52840 to your computer and loading a simple CircuitPython LED demo program (https://adafru.it/DOd) to see the curved infinity mirror in action. Remember to edit the program to account for the 24 LEDs connected to pin D5.

Despite the collar's thinness, the rainbow LED reflections appear to stretch infinitely far beyond the surface, making for a mesmerizing visual effect.

In the next section of this Guide, we'll take advantage of the ItsyBitsy nRF52840's fast, CircuitPython-capable processor and its Bluetooth capabilities to create a variety of fun animations which can be controlled remotely.

CircuitPython Code

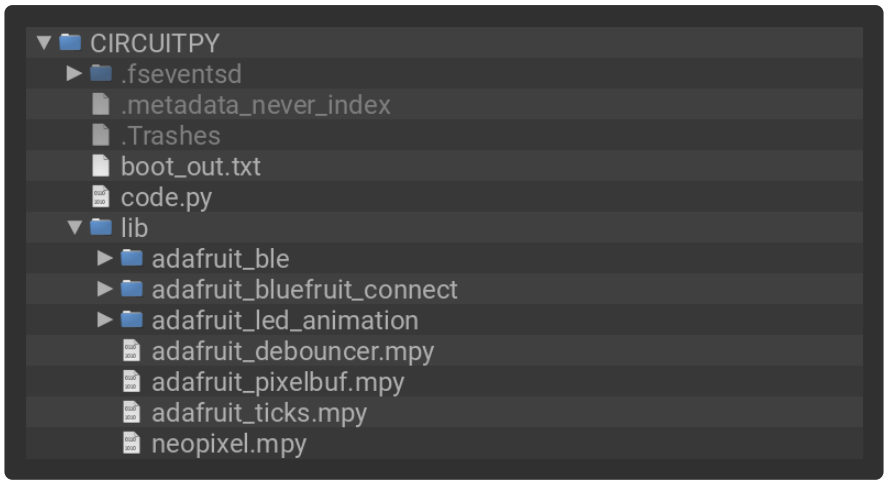
Connect the Collar

Connect the ItsyBitsy nRF52840 to a computer with a micro USB cable to program it. Be sure that your cable carries data as well as power.

Connecting the USB cable will also charge the battery. If you've programmed the ItsyBitsy already, you will notice that the LED patterns run whenever the collar is connected to USB power. The slide switch only turns off the power from the LiPo battery. This occurs because the LEDs connect to the ItsyBitsy's Vhi pin, which draws from both USB and battery power. This feature stops the battery from being drained while you're writing code, but may be annoying if you'd just like to leave your collar plugged in for charging. For this reason, we'll create a "charging" mode in the code that allows us to toggle the animations on and off when the the ItsyBitsy's SW button is pressed.

Install CircuitPython

If you're not familiar with the process, this learn guide (https://adafru.it/Amd) gives a good overview of installing CircuitPython on a number of microcontrollers, including the ItsyBitsy nRF52840.



Load the Libraries

If you don't already have it, download the latest CircuitPython release from the link below. Open the zipped file to get the CircuitPython libraries. Copy the following libraries to the ItsyBitsy folder by dragging them into the **lib** folder in the **CIRCUITPY** drive that appears when you connect the board to your computer.

- adafruit_ble
- adafruit_bluefruit_connect
- adafruit_bus_device
- adafruit_debouncer
- adafruit_led_animation
- adafruit_ticks
- digitalio
- neopixel.mpy

Download the Latest Adafruit CircuitPython Release

https://adafru.it/ENC

The Complete Code

Click download and save on your computer. Plug your ItsyBitsy into your computer via a known, good USB cable. The board should show up as a flash drive named **CIRCUITPY**. Copy the code.py file to that drive's main (root) directory.

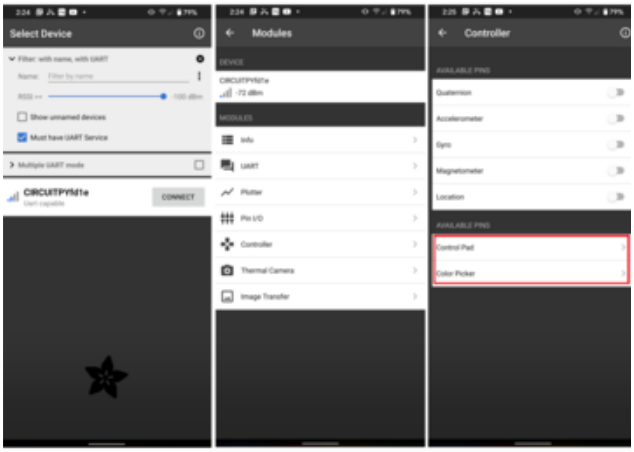
```
# SPDX-FileCopyrightText: 2021 Anne Barela for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""
```

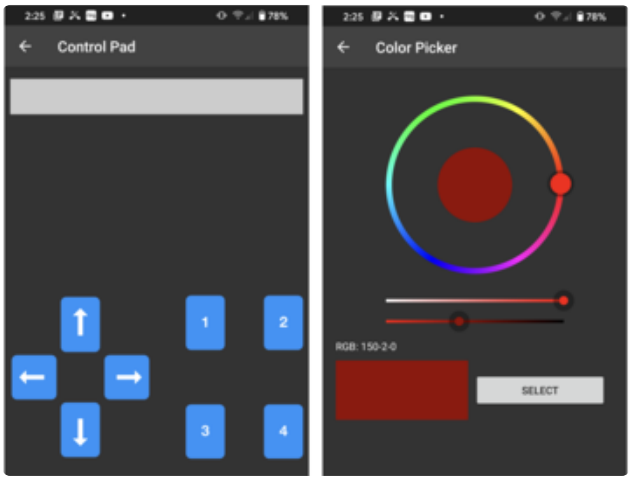
```
Code for the LED Infinity Mirror Collar. Allows the animation sequence
and color to be controlled by input from the Adafruit Bluefruit App
"""
import board
import random
from rainbowio import colorwheel
import neopixel
import digitalio
from adafruit_debouncer import Debouncer

# LED Animation modules
from adafruit_led_animation.animation.comet import Comet
from adafruit_led_animation.animation.chase import Chase
from adafruit_led_animation.animation.rainbowcomet import RainbowComet
from adafruit_led_animation.animation.pulse import Pulse
from adafruit_led_animation.sequence import AnimationSequence


# Bluetooth modules
from adafruit_ble import BLERadio
from adafruit_ble.advertising.standard import ProvideServicesAdvertisement
from adafruit_ble.services.nordic import UARTService
from adafruit_bluefruit_connect.packet import Packet
from adafruit_bluefruit_connect.color_packet import ColorPacket
from adafruit_bluefruit_connect.button_packet import ButtonPacket

# NeoPixel control pin
pixel_pin = board.D5

# Number of pixels in the collar (arranged in two rows)
pixel_num = 24

pixels = neopixel.NeoPixel(pixel_pin, pixel_num, brightness=0.5, auto_write=False)

# Create a switch from the ItsyBity's on-board pushbutton to toggle charge mode
mode_pin = digitalio.DigitalInOut(board.SWITCH)
mode_pin.direction = digitalio.Direction.INPUT
mode_pin.pull = digitalio.Pull.UP
switch = Debouncer(mode_pin)

# Create the animations
comet = Comet(pixels, speed=0.06, color=(180,0,255), tail_length=10, bounce=True)
chase = Chase(pixels, speed=0.05, size=3, spacing=3, color=(0,255,255),
reverse=True)
rainbow_comet = RainbowComet(pixels, speed=.06)
pulse = Pulse(pixels, speed=.04, color=(255,0,0), period = 0.2)


# Our animations sequence
seconds_per_animation = 10
animations = AnimationSequence(comet, rainbow_comet, chase,
advance_interval=seconds_per_animation, auto_clear=True)
# Current display determines whether we are showing the animation sequence or the
pulse animation
current_display = animations

# Mode changes the color of random animations randomly
random_color_mode = True

def random_animation_color(anims):
    if random_color_mode:
        anims.color = colorwheel(random.randint(0,255))

animations.add_cycle_complete_receiver(random_animation_color)


# After we complete three pulse cycles, return to main animations list
def pulse_finished(anim):
    global current_display
```

```
    current_display = animations

pulse.add_cycle_complete_receiver(pulse_finished)
pulse.notify_cycles = 3


# Bluetooth
ble = BLERadio()
uart_service = UARTService()
advertisement = ProvideServicesAdvertisement(uart_service)

# Set charge_mode to True to turn off the LED Animations and Bluetooth
#  e.g. when charging the battery
charge_mode = False

# Checks the ItsyBitsy's switch button
def check_switch():
    global charge_mode
    switch.update()
    if switch.fell:  #Switch changed state
        charge_mode = not charge_mode
        # if display has just been turned off, clear all LEDs, disconnect, stop
advertising
        if charge_mode:
            pixels.fill((0,0,0))
            pixels.show()
            if ble.connected:
                for conn in ble.connections:
                    conn.disconnect()
            if ble.advertising:
                ble.stop_advertising()

# Main program loop
while True:
    # Check whether charge mode has been changed
    check_switch()
    if charge_mode:
        pass
    else:
        current_display.animate()
        if ble.connected:
            if uart_service.in_waiting:
                #Packet is arriving
                packet = Packet.from_stream(uart_service)
                if isinstance(packet, ButtonPacket) and packet.pressed:
                    if packet.button == ButtonPacket.BUTTON_1:
                        # Animation colors change to a random new value after every
animation sequence
                        random_color_mode = True
                    elif packet.button == ButtonPacket.BUTTON_2:
                        # Animation colors stay the same unless manually changed
                        random_color_mode = False
                    elif packet.button == ButtonPacket.BUTTON_3:
                        # Stay on the same animation
                        animations._advance_interval = None
                    elif packet.button == ButtonPacket.BUTTON_4:
                        # Auto-advance animations
                        animations._advance_interval = seconds_per_animation*1000
                    elif packet.button == ButtonPacket.LEFT:
                        # Go to first animation in the sequence
                        animations.activate(0)
                    elif packet.button == ButtonPacket.RIGHT:
                        # Go to the next animation in the sequence
                        animations.next()
                elif isinstance(packet, ColorPacket):
                    animations.color = packet.color
                    pulse.color = packet.color
                    # temporarily change to pulse display to show off the new color
                    current_display = pulse
```

```
        else:
            if not ble.advertising:
                ble.start_advertising(advertisement)
```

Bluefruit Control





Install the Bluefruit app on your phone or tablet to control the collar remotely.

When you open the app, the initial screen allows you to select and connect to the the collar. Once connected, select the "Controller" option to be able to change the display using the "Control Pad" and "Color Picker" settings.

The Control Pad has four arrow buttons and four number buttons. Their functions are:

left arrow resets the display to the first animation in the sequence
right arrow moves to the next animation
button 1 selects a mode in which the animation colors change to a random value in each cycle. This mode is the default
button 2 sets a mode where the colors stay the same unless manually changed with the color picker.
button 3 sets the animation sequence to switch between animations at a regular interval. This is the default mode.
button 4 sets the animation sequence to remain on the current animation unless manually changed with the left/right arrow buttons

Code Features and Functionality

The complete code to create Bluetooth-responsive animations on the Infinity Mirror Collar can be found at the bottom of this page. This section will examine how specific features are implemented in the CircuitPython code.

Controlling LED Animations

Writing LED animation code that runs smoothly while switching patterns and responding to button and Bluetooth inputs can be a challenge. Fortunately, the **adafruit_led_animation** library has a simple interface which handles animation timing and sequencing behind the scenes. This learn guide (https://adafru.it/LZF) provides a great overview of how to use the library to easily generate a number of great-looking, customizable sequences.

All you need to know to use the library is how to create different animations, and place them in a sequence. The first step is to import the modules for the required animations, as shown below.

```
# LED Animation modules
from adafruit_led_animation.animation.comet import Comet
from adafruit_led_animation.animation.chase import Chase
from adafruit_led_animation.animation.rainbowcomet import RainbowComet
from adafruit_led_animation.animation.pulse import Pulse
from adafruit_led_animation.sequence import AnimationSequence
from adafruit_led_animation.color import colorwheel
```

Once the modules are imported, we'll instantiate different animation objects with various parameters, such as color and speed, that determine their appearance.

The code below creates four animation objects. The first three animations run repeatedly in sequence, and the last one, **pulse**, displays briefly when the user selects a new color with the Bluefruit App.

The code below creates an `AnimationSequence` that automatically takes care of running and transitioning between the `comet`, `rainbow_comet`, and `chase` animations we've created.

```
# Create the animations
comet = Comet(pixels, speed=0.06, color=(180,0,255), tail_length=10, bounce=True)
chase = Chase(pixels, speed=0.05, size=3, spacing=3, color=(0,255,255),
reverse=True)
rainbow_comet = RainbowComet(pixels, speed=.06)
pulse = Pulse(pixels, speed=.04, color=(255,0,0), period = 0.2)

# Our animations sequence
seconds_per_animation = 10
```

```
animations = AnimationSequence(comet, rainbow_comet, chase,
advance_interval=seconds_per_animation, auto_clear=True)
```

Even though we set the colors of comet and chase animation objects when we created them, we can still change them afterwards, by setting their color property. In the code, we define a `random_color_mode` variable which, when set to True, changes the animation colors in each cycle.

The `AnimationSequence` object allows us to register a function which will run after each complete animation cycle using its `add_cycle_complete_receiver()` method. The code below defines a function that, when `random_color_mode` is enabled, changes the `comet` and `chase` animation colors to a new, randomly selected value at the end of every cycle.

```
# Mode changes the color of random animations randomly
random_color_mode = True

def random_animation_color(anims):
    if random_color_mode:
        anims.color = colorwheel(random.randint(0,255))

animations.add_cycle_complete_receiver(random_animation_color)
```

Displaying an Animation out of Sequence

The code signals that it has received input from the Bluefruit App color picker by setting the animation colors and displaying a pulse animation in the newly selected color.

Changing the default colors of all of the animations is a simple matter of changing their color property. The animation sequence automatically propagates that change to all of its child animations.

The code switches between the animation sequence and the pulse animation by creating a variable, named `current_display`, which holds whatever animation or animation sequence is currently being shown. It is initially set to display the animation sequence.

```
# Current display determines whether we are showing the animation sequence or the
pulse animation
current_display = animations
```

Since both animation and animation display objects have an `animate()` method, a single function call can run either animation in the main loop.

```
current_display.animate()
```

When new color information is received, the default animation colors are changed and the `current_display` variable is set to the pulse animation.

```
elif isinstance(packet, ColorPacket):
                    animations.color = packet.color
                    pulse.color = packet.color
                    # temporarily change to pulse display to show off the new color
                    current_display = pulse
```

The pulse animation will automatically set the display back to the main animation sequence when it finishes, because we have registered a special function that it will run after every three cycles.

```
# After we complete three pulse cycles, return to main animations list
def pulse_finished(anim):
    global current_display
    current_display = animations

pulse.add_cycle_complete_receiver(pulse_finished)
pulse.notify_cycles = 3
```

Bluetooth Functionality

The BLE code checks for data packets and processes any button or color packets it receives in the main loop. If it is not connected it will start advertising so that it is available for a connection from the Bluefruit app.

```
if ble.connected:
            if uart_service.in_waiting:
                #Packet is arriving
                packet = Packet.from_stream(uart_service)
                if isinstance(packet, ButtonPacket) and packet.pressed:
                    if packet.button == ButtonPacket.BUTTON_1:
                        # Animation colors change to a random new value after every
animation sequence
                        random_color_mode = True
                    elif packet.button == ButtonPacket.BUTTON_2:
                        # Animation colors stay the same unless manually changed
                        random_color_mode = False
                    elif packet.button == ButtonPacket.BUTTON_3:
                        # Stay on the same animation
                        animations._advance_interval = None
                    elif packet.button == ButtonPacket.BUTTON_4:
                        # Auto-advance animations
                        animations._advance_interval = seconds_per_animation*1000
                    elif packet.button == ButtonPacket.LEFT:
                        # Go to first animation in the sequence
                        animations.activate(0)
                    elif packet.button == ButtonPacket.RIGHT:
                        # Go to the next animation in the sequence
                        animations.next()
                elif isinstance(packet, ColorPacket):
                    animations.color = packet.color
                    pulse.color = packet.color
                    # temporarily change to pulse display to show off the new color
                    current_display = pulse

        else:
```

```
            if not ble.advertising:
                ble.start_advertising(advertisement)
```

Charging Mode

As mentioned earlier, we don't want the LEDs to run if we've only connected the collar to power to charge it. The code contains a `charge_mode` variable which, when set to `True`, disables Bluetooth connections and turns off the display. Charge mode is toggled on and off by pressing the SW button on the Itsy Bitsy nRF5840.

```
# Set charge_mode to True to turn off the LED Animations and Bluetooth
#  e.g. when charging the battery
charge_mode = False
```

The CircuitPython **adafruit_debouncer** module, used in the code below lets us easily track and respond to button presses.

```
# Create a switch from the ItsyBity's on-board pushbutton to toggle charge mode
mode_pin = digitalio.DigitalInOut(board.SWITCH)
mode_pin.direction = digitalio.Direction.INPUT
mode_pin.pull = digitalio.Pull.UP
switch = Debouncer(mode_pin)
```

The Debouncer module removes the uncertainty as to whether a button press is real or an artifact of voltage fluctuations on a pin. You can learn more about debouncing a switch pin here (https://adafru.it/lif). Our code attaches a debouncer object to the ItsyBitsy's **SWITCH** pin, allowing us to easily track when the state of the pin has changed due to the button being pressed or released.

In each cycle of the main code loop, we call the function `check_switch()`, shown below, which updates our debouncer and tracks and responds to changes in the pin state. When set to charging mode, the code turns off all the LEDs, disconnects any BLE connections, and turns off BLE advertising.

```
# Checks the ItsyBitsy's switch button
def check_switch():
    global charge_mode
    switch.update()
    if switch.fell:  #Switch changed state
        charge_mode = not charge_mode
        # if display has just been turned off, clear all LEDs, disconnect, stop
advertising
        if charge_mode:
            pixels.fill((0,0,0))
            pixels.show()
            if ble.connected:
                for conn in ble.connections:
                    conn.disconnect()
            if ble.advertising:
                ble.stop_advertising()
```