DotStar Fortune Necklace with Bluetooth and Touch

Created by Charlyn G

https://learn.adafruit.com/dotstar-fortune-necklace

Last updated on 2024-03-08 03:41:03 PM EST

Table of Contents

Overview 3

- Geometric LED matrix pendant with fortune-telling talent
- Parts for Pendant
- Parts for Electronics + Power

Circuit Diagrams 7

- ItsyBitsy and Power Circuit
- Necklace LED Matrix Circuit

3D printing or laser cutting 10

Pendant wiring and assembly 11

ItsyBitsy + Power wiring 20

CircuitPython for ItsyBitsy nRF52840 Express 25

- Set up CircuitPython Quick Start!
- Further Information

CircuitPython code 27

- Installing Project Code
- Test capacitive touch
- Test Bluefruit app interaction

Final Assembly 35

- 3D printed hex case
- Laser cut rectangular case

Overview

Geometric LED matrix pendant with fortune-telling talent

Sometimes, you want to be subtle, and sometimes you want to be loud. And sometimes, you just want to ask the Fortunes a simple yes or no question.

Now you can do all that with this modern necklace that looks subtle and simple at first glance, and then turns into a versatile light show with a touch or a tap. It even reaches into the random number universe to answer your categorical questions when you touch it!

This necklace is programmed with CircuitPython and made with the adorable DotStar LED Matrix, powered by the ItsyBitsy nRF52840 so you can easily customize your style over Bluetooth using the Adafruit Bluefruit Connect App!

The front panel is a fascinating black LED acrylic material, which stays opaque until LED lights shine through. It also has a brass piece that acts as a capacitive touch button and a rechargeable battery thanks to the LiPo backpack.

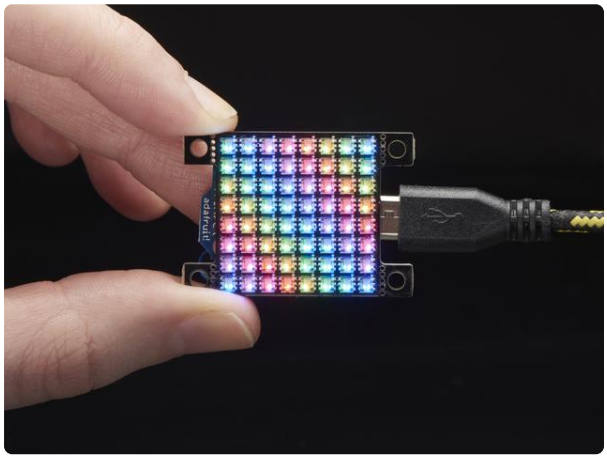
The ItsyBitsy and LiPo battery are all housed inside a case that can be worn behind your neck, and the necklace is connected via a USB micro-B connector, so you can change the electronics housing if you desire. There is a lot of room for customization here to really make this project your own!

This project requires quite a bit of soldering, so here are some excellent guides if you want to brush up on the basics:

- Adafruit Guide To Excellent Soldering (https://adafru.it/drl)
- Collin's Lab: Soldering (https://adafru.it/dyT)

You'll also need to either laser cut or 3D print the both the pendant frame and electronics case. The black LED acrylic can be milled or laser cut to a simple square shape.

Parts for Pendant



Adafruit DotStar High Density 8x8 Grid - 64 RGB LED Pixel Matrix
Do not eat this LED grid just because it is so colorful and bite-sized! This is the tiniest little LED grid we could make, with 64 full RGB color pixels in a square that is only...
https://www.adafruit.com/product/3444

Black LED Diffusion Acrylic Panel - 10.2" x 5.1"

nice whoppin' rectangular slab of some lovely black acrylic to add some extra diffusion to your LED Matrix project. This material is 2.6mm (0.1") thick and is made of...

https://www.adafruit.com/product/4749

1 x Silicone Stranded-Core Ribbon Wire - 10 Wire
For easy neat wiring
https://www.adafruit.com/product/3890

1 x Brass kite shape
For capacitive touch sensing (choose whatever shape you like!)
https://www.etsy.com/listing/760890902/12-flat-brass-diamond-stamping-blanks

1 x Small zip tie
Secure the wires on top of the pendant
https://amzn.to/37YS6QC

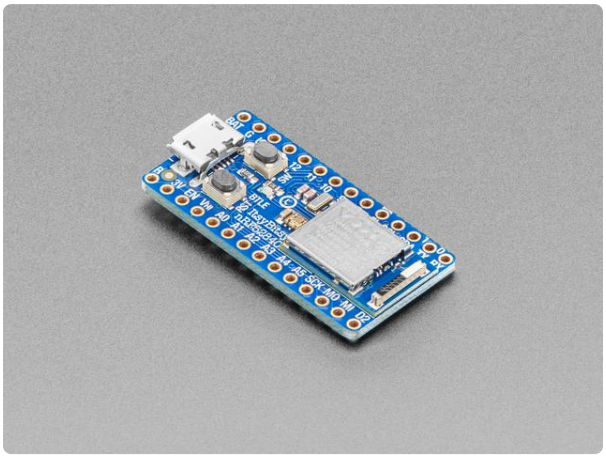
1 x Double sided foam tape
Strong enough to stick to acrylic and PCBs

1 x Super glue
Get the gel kind so you have a few more seconds before it dries

1 x Jump ring
Optional. Makes it easy to attach the pendant to some thread
https://amzn.to/3qTfhUX

1 x Enameled copper wire
Optional. Use as a thin wire to connect the brass shape to the board.
https://amzn.to/38lYAsP

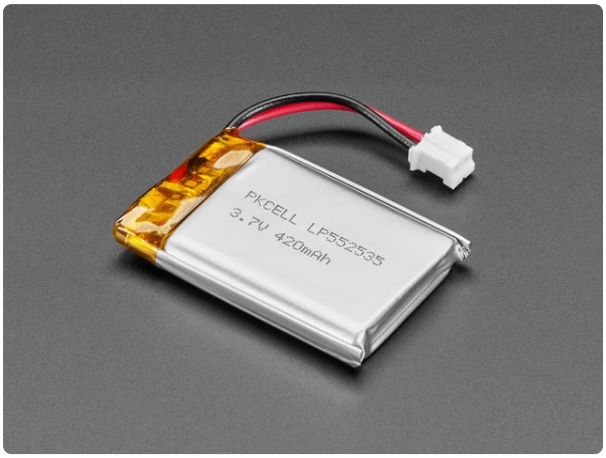
Parts for Electronics + Power



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



Lithium Ion Polymer Battery with Short Cable - 3.7V 420mAh

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/4236

1 x USB DIY Slim Connector Shell - MicroB Plug https://www.adafruit.com/product/1826

To connect the necklace to the ItsyBitsy and power

---

1 x LiIon/LiPoly Backpack Add-On https://www.adafruit.com/product/2124

Power the ItsyBitsy via a rechargeable battery

---

1 x Slide Switch https://www.adafruit.com/product/805

SPDT switch to control power when battery is connected

---

1 x Embroidery threads https://amzn.to/2VMG8UA

Gold thread to wrap necklace cord and allow braiding of wires

---

1 x 1M (or greater) ohm resistor

For the capacitive touch pin

---

1 x Double sided thin tape

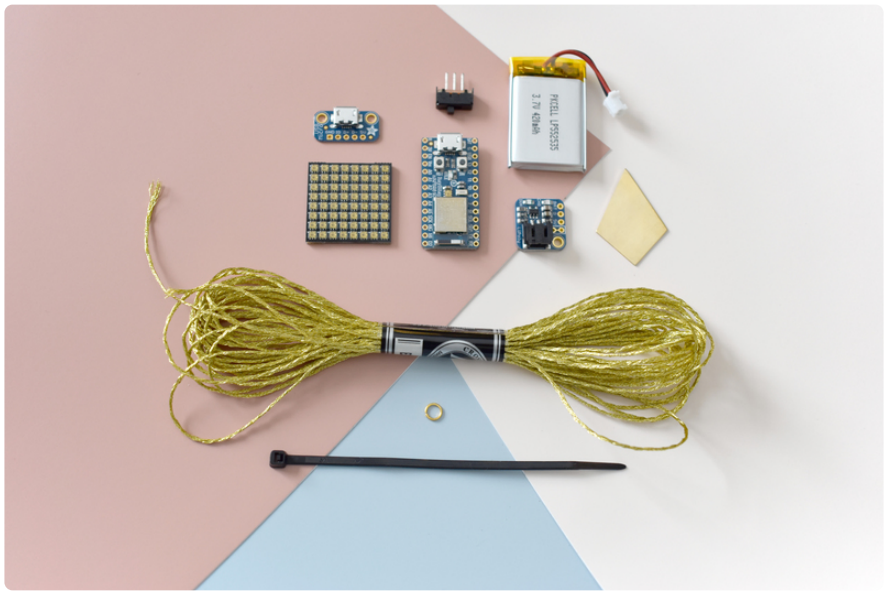
For securing the battery

Optional. To decorate the electronics case with!

1 x Brass teardrop shape

Optional. To decorate the electronics case with!

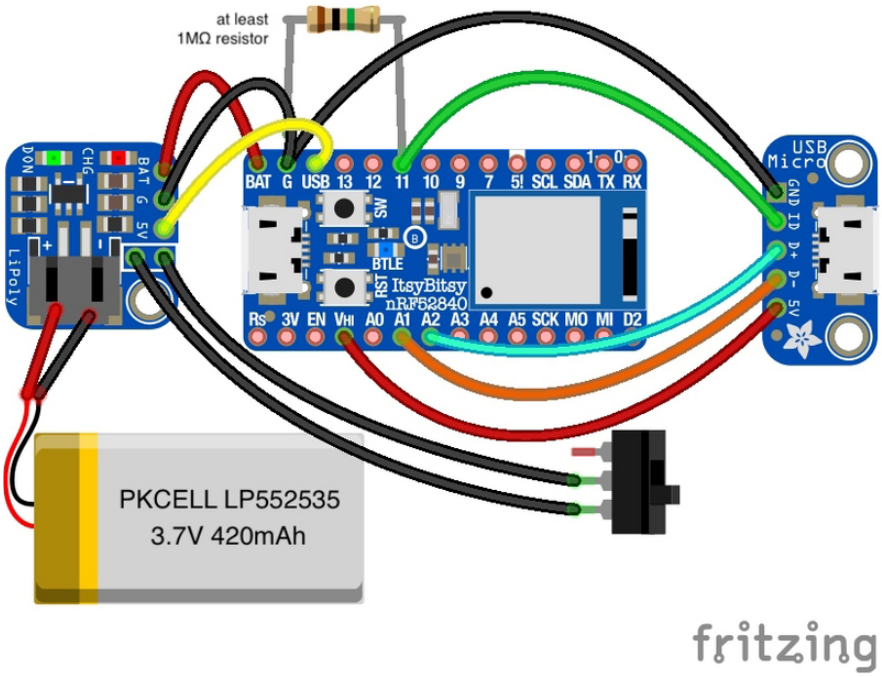
https://www.etsy.com/listing/178463091/18-small-flat-brass-teardrop-stamping



Circuit Diagrams

There are two circuits to build: the necklace itself and the ItsyBitsy + power housed inside the case. These are connected via the USB MicroB counterparts. These diagrams will detail which circuit connections need to be made, but we'll go over assembly in more detail later.

ItsyBitsy and Power Circuit



Note that the diagram above doesn't show the exact placement of the components -- it was laid out so that you can easily see the wire connections. Also note that this wiring means that the slide switch will turn off battery power, but it has no effect if the ItsyBitsy is connected to USB.

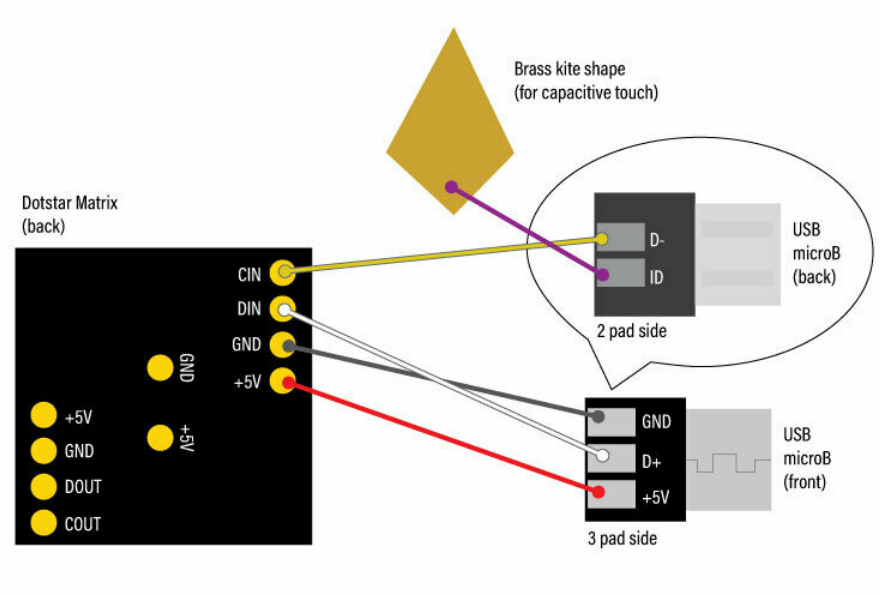
Here is a table for the connections illustrated above to help make it more clear:

LiPo backpack	ItsyBitsy	USB micro-B
BAT	BAT	-
G	G	G
5V	USB	-
-	VHi	5V
-	A1	D-
-	A2	D+
-	D11	ID

To make the slide switch a power switch, cut the copper trace between the two through-hole connectors on the **LiPo backpack** next to the **5V** hole, and solder two wires through those to the slide switch.

To support capacitive touch on **D11**, connect a **1MΩ** resistor from **D11** to Ground.

Necklace LED Matrix Circuit



The above diagram is a representation of the circuit connections that need to be made between the DotStar 8x8 Matrix and the USB MicroB plug. Note that there is a single MicroB plug pictured, with both of its sides showing.

DotStar Matrix	USB MicroB plug
CIN	D-
DIN	D+
GND	GND
+5v	+5V
-	ID (to brass kite)

3D printing or laser cutting

You can choose either laser cut parts or 3d printed parts for this project. Either way though, you'll need to cut the Black LED Acrylic front piece to the exact size needed. The cut dimensions of the acrylic varies depending on what method you choose:

- If using the laser cut frame, cut the acrylic to **28.73mm x 28.70mm** (the PDF and DXF files include this)
- If using the 3d printed frame, cut the acrylic to **29.02mm x 28.99mm**

The button link below includes both the laser cutter files and the 3d printer files:

Download Pendant parts on Thingiverse

https://adafru.it/Poc

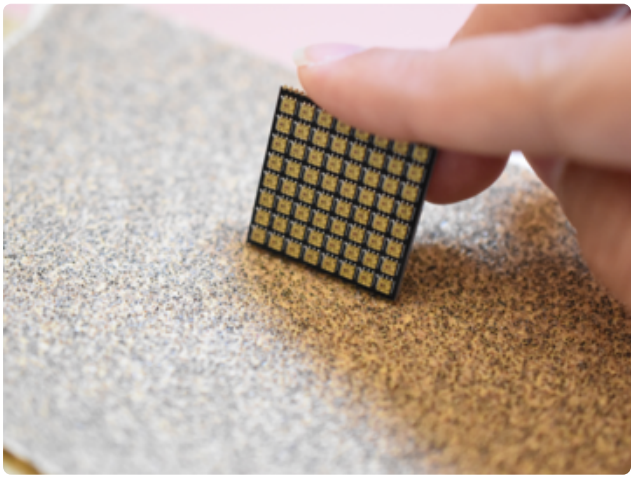
You can also choose between two different types of cases that will hold the electronics and the battery. The hexagonal 3D printed case is a remix of the case from Neopixel LED Heart Necklace (https://adafru.it/Pod) and is quite a bit easier to put together. The laser cut rectangular case will be faster to cut out but requires a bit of glueing. Choose the shape that most pleases you!

The button link below also includes both the laser cutter files and the 3d printer files:

Download Case parts on Thingiverse

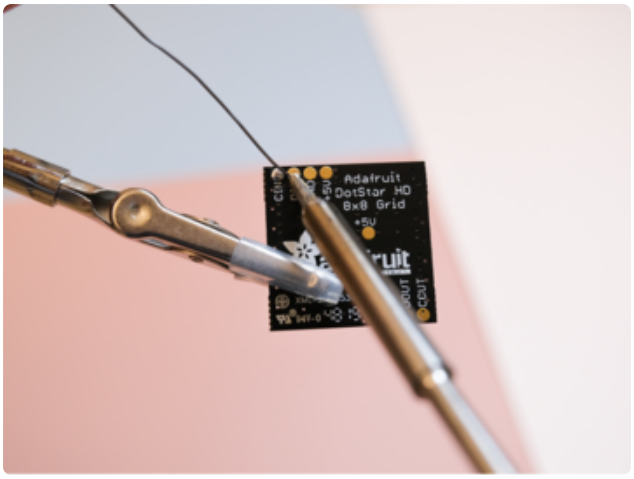
https://adafru.it/Poe

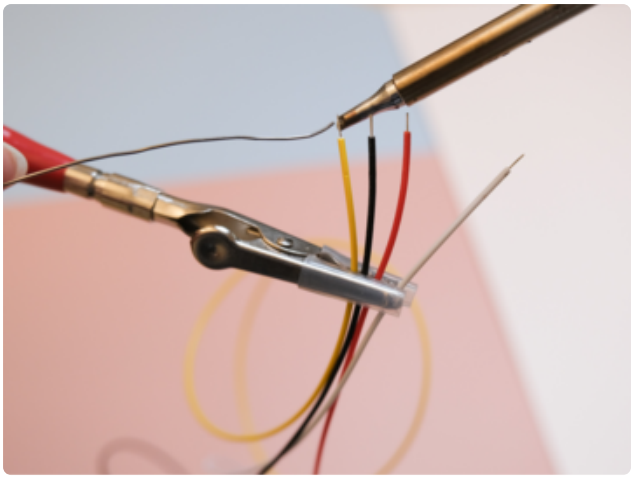
Pendant wiring and assembly

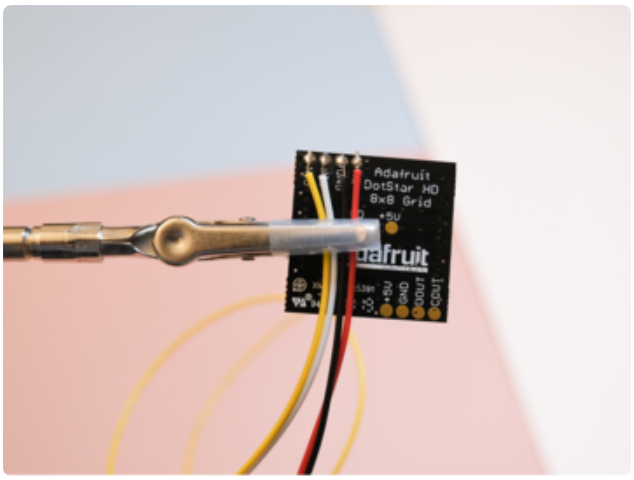


Prepare the DotStar Matrix

Cut the mounting tabs off the DotStar matrix and carefully sand down with coarse grit sandpaper to smooth the edges. Test if it press fits into the pendant frame you chose before sanding down any further -- you don't want it too loose.





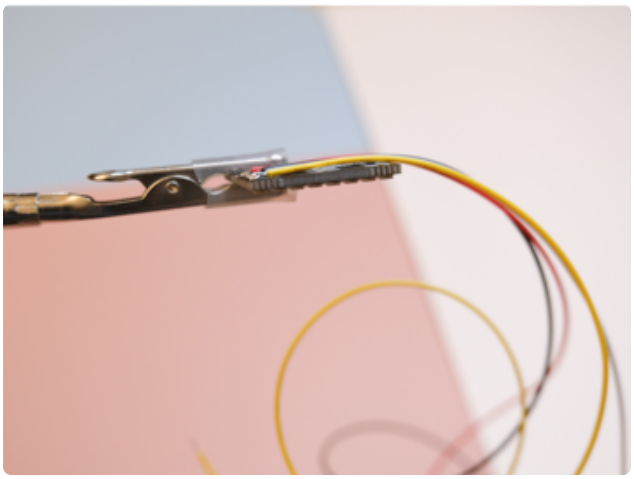


Solder wires

The length of the wires will depend on the diameter of your head since they'll need to be long enough to fit your head through. **20cm** should be a good starting point, adjust as needed.

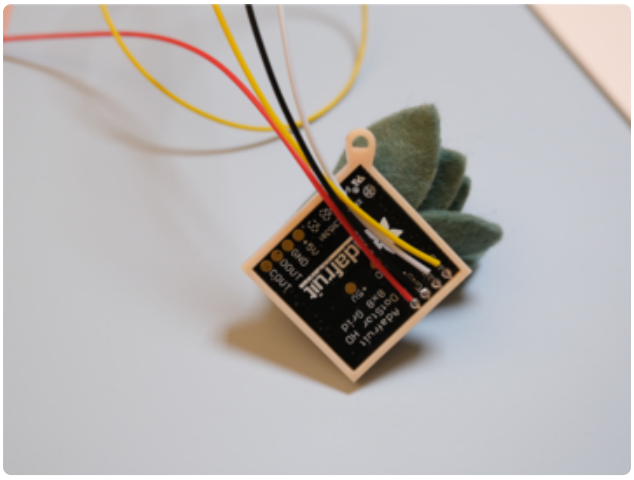
Tin the pads for all 4 contacts first, and then tin the exposed wires as well before soldering together. Make sure to cut any leads sticking out after.

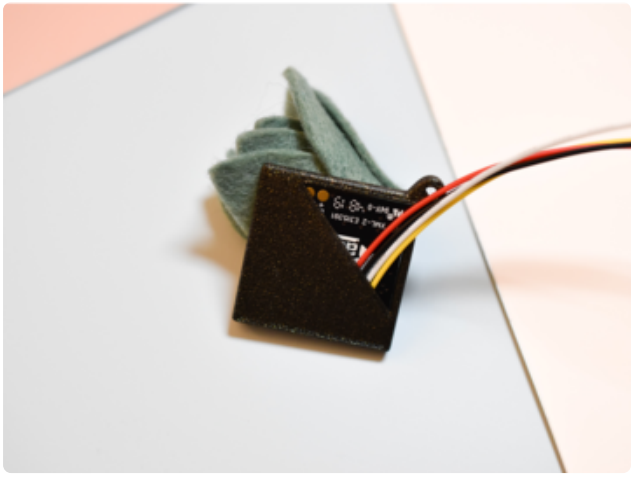
It will help to use a different color wire for each pad, so that you can easily identify which wire is which later.



Make sure that the wires are flat

Double check that the wires are soldered as flat as possible against the PCB so that it will fit nicely within the pendant frame.





Attach matrix to the frame

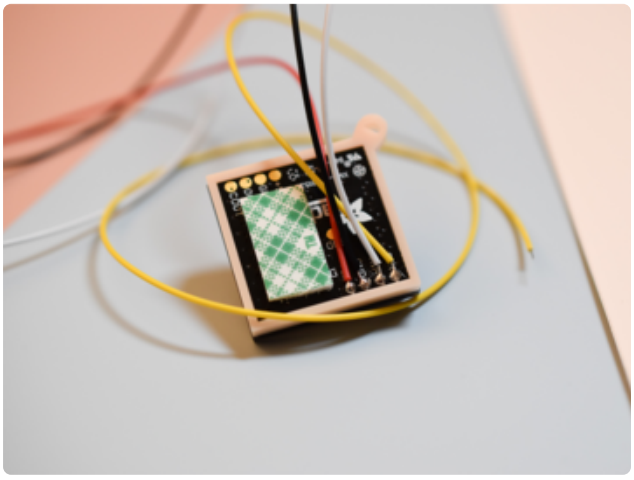
Using either the 3d printed frame or the laser cut frame, push the wired matrix to the frame, it should just press fit in.

You want the wires to be oriented as shown -- if the top hole is "north", the wires should be "southeast" of the hole. This orientation will ensure that the back cover will protect the wires AND it is the orientation that the code animations are setup for.

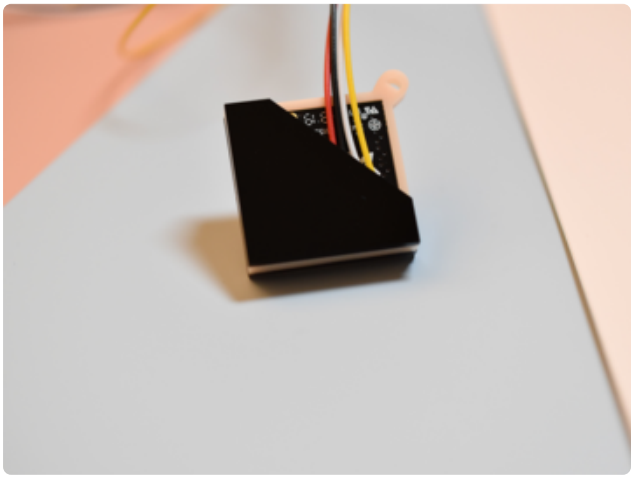
Glue the top cover to the frame

Using super glue or any other appropriate glue, secure the black LED front cover so that it's aligned with the frame and covers the LED matrix entirely.

Tip: using the "gel" kind for super glue will allow you to have significantly more seconds to align the cover to the frame before the glue dries.



Secure the back cover



For laser cut frame only. Secure the back cover using foam double sided tape.

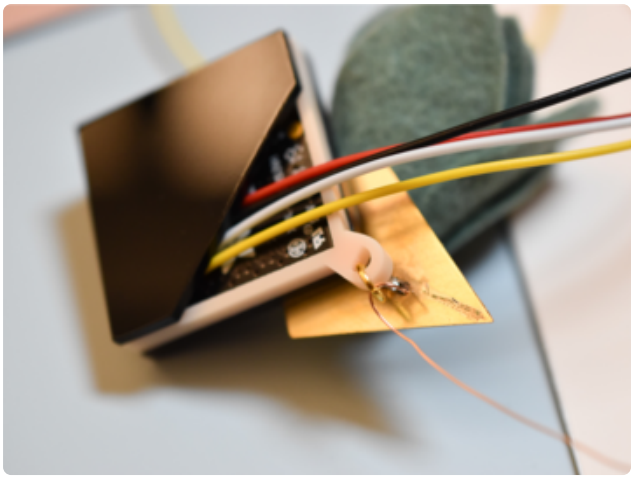
Solder a wire to the brass shape

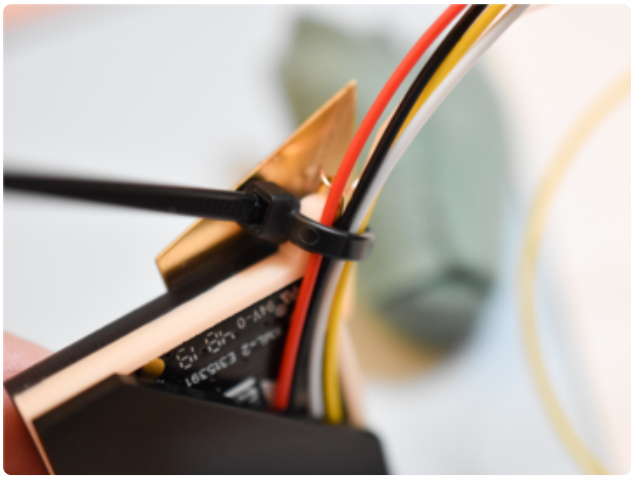
This brass shape will be used as a capacitive touch button. You can use the same wires as the ones used for the matrix, but that will add a bit of bulk to the necklace, so here I've opted to use some thin enameled copper wire instead.

Make sure that you're using a heat resistant silicone surface when soldering the wire to brass like this since the brass piece will get really hot. Check continuity with a multimeter to make sure that the solder joint is good.

Secure the brass shape onto the cover

Use some glue or double sided tape, and adjust to your desired position. Pause to admire the beauty that you have created so far :)



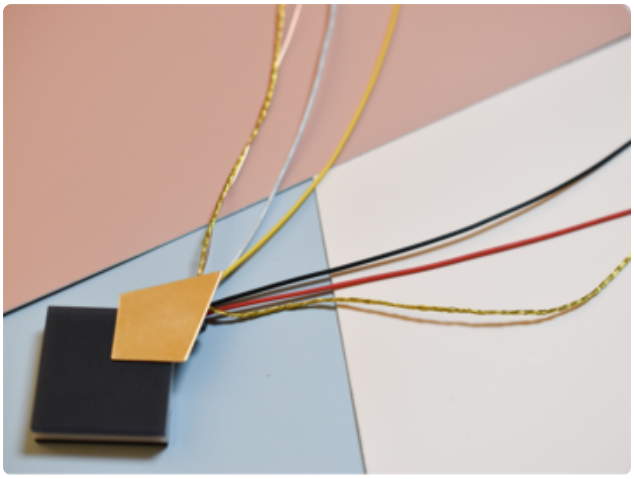


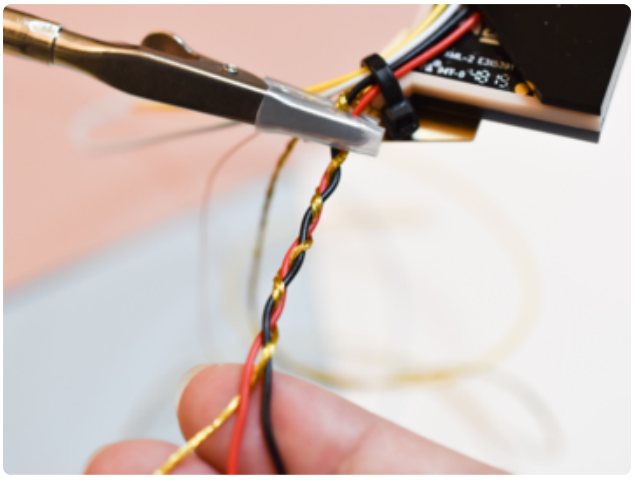
Secure the wires and use embroidery floss for support

Optionally attach a jump ring through the hole, then use a small zip tie to secure the wires together at the top of the pendant.

Then, tie a piece of embroidery floss either directly through the pendant hole or through the jump ring if you attached one. This will help give the necklace some support, and will allow you to braid the wires together.

The length of the resulting 2 halves of the embroidery thread should be slightly longer than the rest of the wires to account for the braiding in the next step.



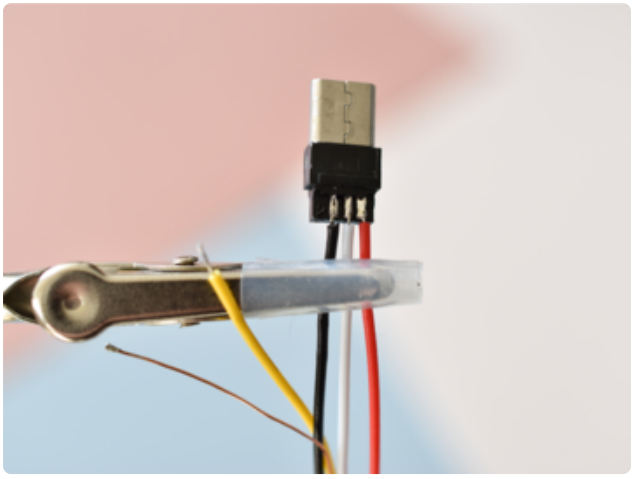


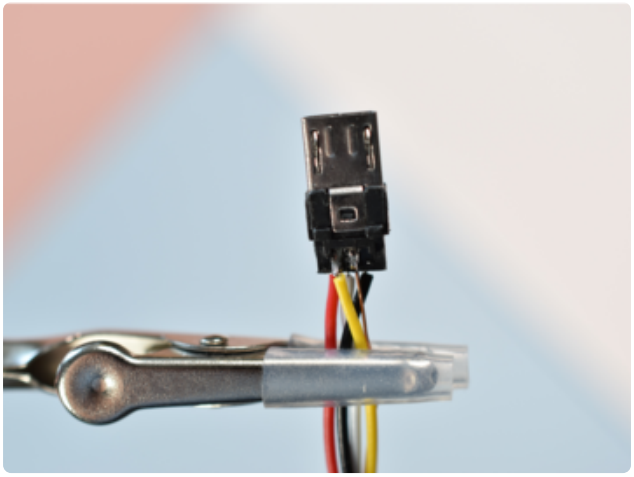
Braid the wires together to form a necklace chain

Separate the wires and the embroidery floss into two groups. Each group should have either 3 or 4 wires + floss each. Braid each group together to form the necklace chain.

After braiding, secure the end with a simple knot of the embroidery floss. This will be a good time to double check that the braided necklace chain will be long enough such that your head fits through.

At this point, you'll have something that looks like the picture below. Pause here to admire your handy work! :)





Solder the necklace chain wires to the USB microB plug

Now that you have double checked that the length of the wires is enough to fit your head through (you did, right?), you can proceed to soldering the ends to the microB plug. First, insert the wires through the two connector pieces as shown. Then, refer to the circuit diagram in the previous page to solder the proper wires. Use the different wire colors to guide you through this step.

Note that the circuit diagram refers to both sides of the microB plug (the 2-pad and 3-pad side)

Optional: Wrap the necklace chain with gold embroidery floss

This step is optional because it takes a long time to do, but the result is quite satisfying! It helps if you wrap the embroidery floss around something cylindrical so that you can more easily maneuver the floss around the braided wires.

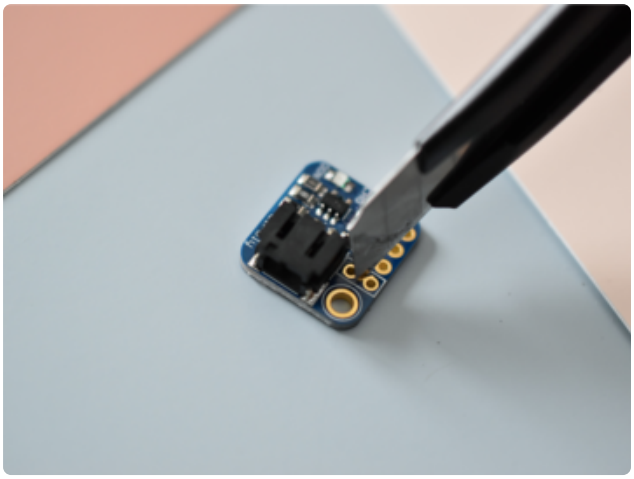
Now, you're ready to tackle the next part: the ItsyBitsy wiring with the LiPo battery!

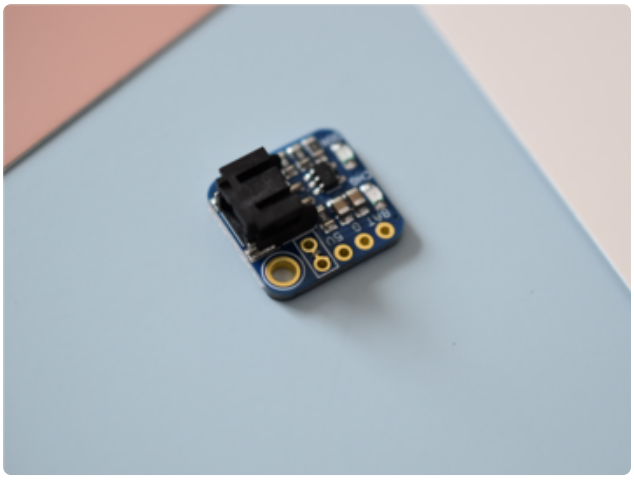
ItsyBitsy + Power wiring

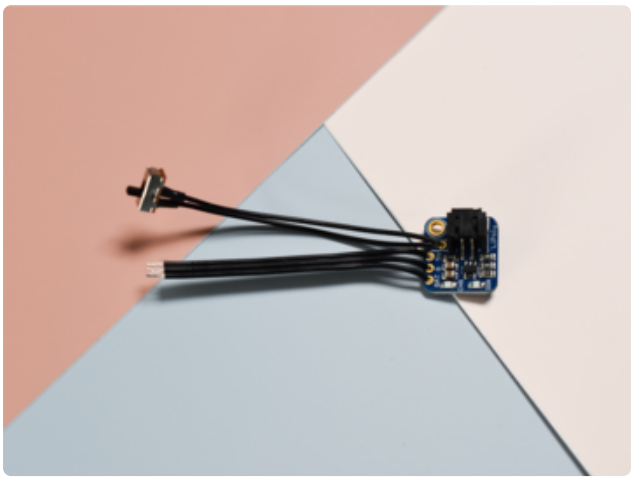
Note: The following photos will show a silver slide switch, but the same steps apply to the Adafruit black slide switches.

Prepare the power slide switch

Cut one of the side legs, and shorten the remaining two legs. Tin the legs, tin the wires, and solder them together. Cover with heat shrink tubing.



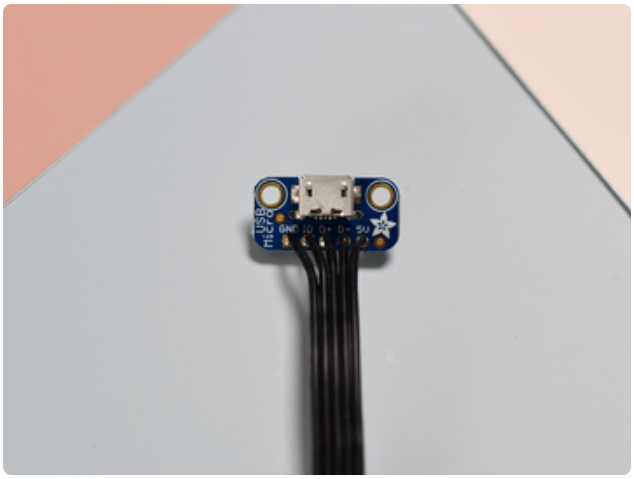




Prepare the LiPo backpack

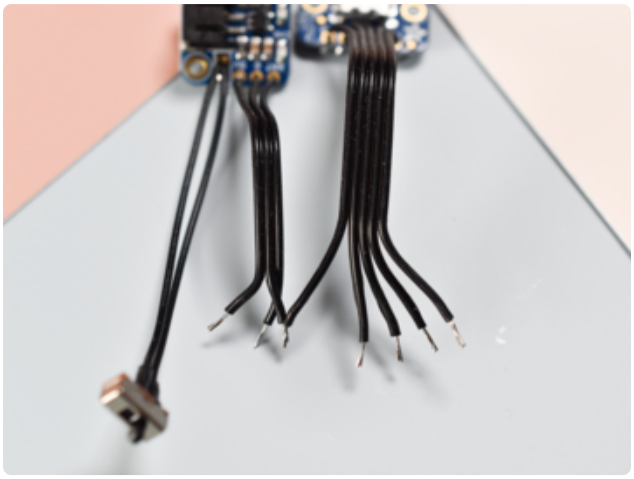
Cut the copper trace between the two holes next to the mounting hole. This will allow us to attach the slide switch that will act as a power switch.

Solder 3 ribbon wires to the through-holes for 5V, BAT and GND. The length of the ribbon wires can be about **5cm**.



Prepare the USB MicroB port

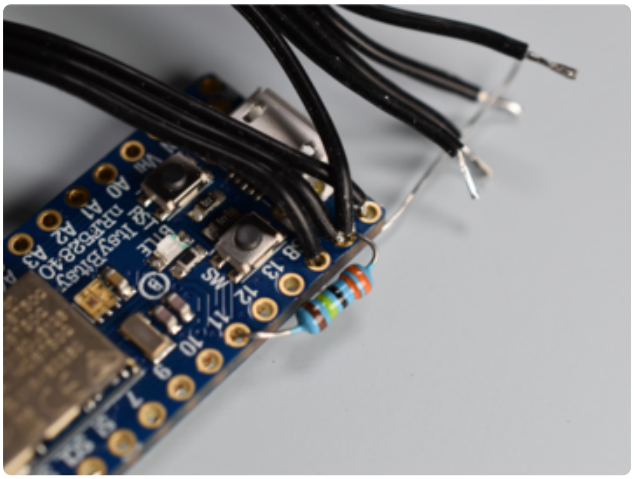
Solder 5 ribbon wires onto all 5 through-holes. The length of the ribbon wires can be around **6cm.**



Combine the ground wires

Carefully identify the GND wires for both the USB microB port and the LiPo backpack, twist them together, and dab a bit of solder to secure.

Solder combined GND wires plus resistor to ItsyBitsy

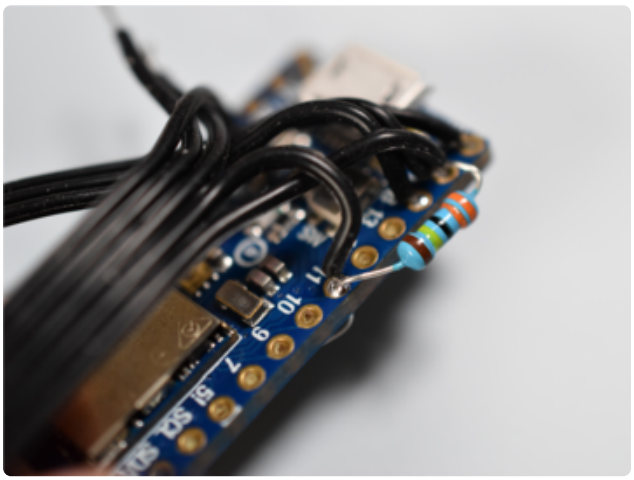


Take the **combined GND** wires and fit them into **GND** of the ItsyBitsy. Before soldering, fit a **1M** resistor or higher into **GND** as well, and put the other end of the resistor through **D11**.

Do not solder on D11 yet.

BAT and **5V** from the LiPo backpack connects to **BAT** and **USB** of the ItsyBitsy respectively. Solder these connections.

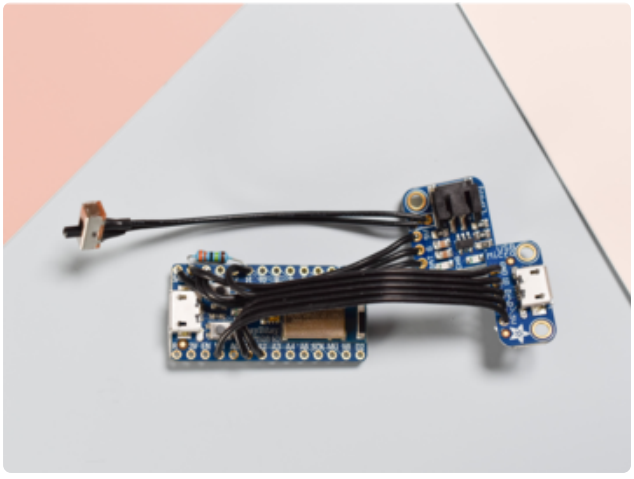
Solder USB MicroB's ID to ItsyBitsy's D11



Carefully identify which wire from the MircoB port is connected to **ID**, and solder that into the ItsyBitsy's **D11** together with the resistor.

This resistor is important to be able to use the brass shape as a capacitive touch button.

Solder remaining 3 wires from microB port



Carefully identify the wires, and connect:

microB port's **5V** to ItsyBitsy's **VHi**
microB port's **D-** to ItsyBitsy's **A1**
microB port's **D+** to ItsyBitsy's **A2**
Once all the wires are soldered, connect the battery to the LiPo backpack and test to see that the green light on the ItsyBitsy turns on.

Now, turn off your soldering iron, find some fresh air away from any lingering fumes, and take a deep breath. You've come quite far in this journey, I'm proud of you. :D

Before we move on with final case assembly, we should setup and load the CircuitPython code into the ItsyBitsy so that we can test and make sure that all our soldered connections work before we stuff everything inside an enclosure!

CircuitPython for ItsyBitsy nRF52840 Express

CircuitPython (https://adafru.it/tB7) is a derivative of MicroPython (https://adafru.it/BeZ) designed to simplify experimentation and education on low-cost microcontrollers. It makes it easier than ever to get prototyping by requiring no upfront desktop software downloads. Simply copy and edit files on the **CIRCUITPY** drive to iterate.

Set up CircuitPython Quick Start!

Follow this quick step-by-step for super-fast Python power :)

Download the latest version of CircuitPython for this board via CircuitPython.org

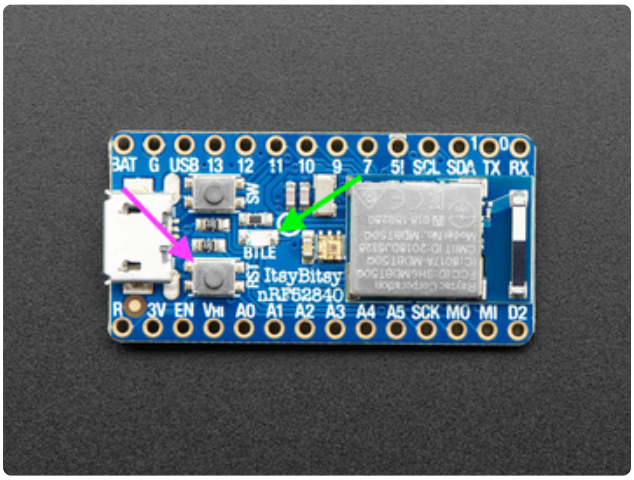
https://adafru.it/Ie7

Further Information

For more detailed info on installing CircuitPython, check out Installing CircuitPython (https://adafru.it/Amd).

Click the link above and download the latest UF2 file.

Download and save it to your desktop (or wherever is handy).



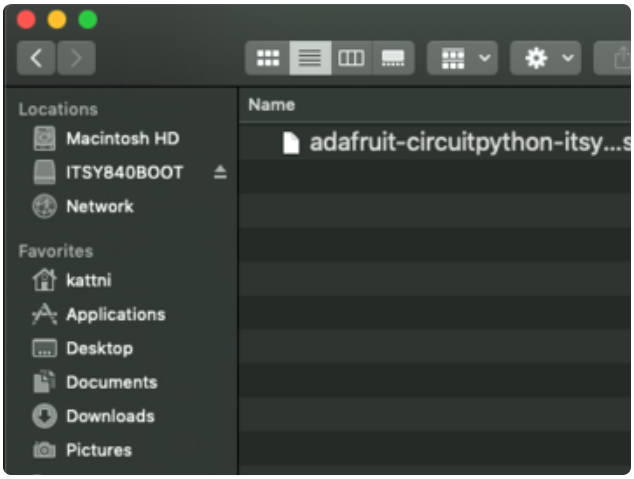
Plug your Itsy nRF52840 into your computer using a known-good USB cable.

A lot of people end up using charge-only USB cables and it is very frustrating! So make sure you have a USB cable you know is good for data sync.

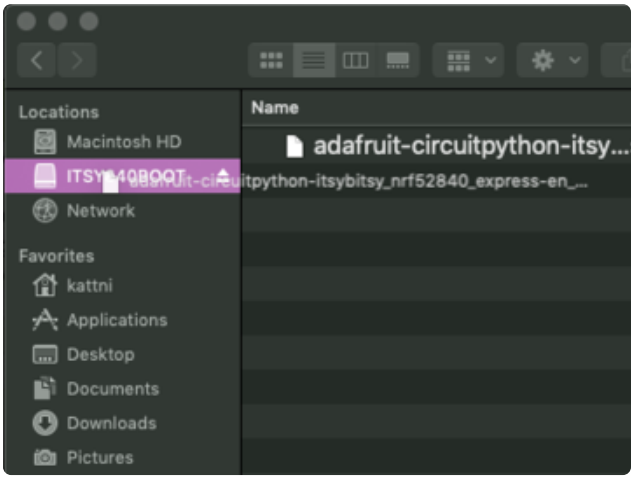
In the image, the **Reset** button is indicated by the magenta arrow, and the **BTLE status LED** is indicated by the green arrow.

Double-click the **Reset** button on your board (magenta arrow), and you will see the **BTLE LED** (green arrow) will pulse quickly then slowly blue. If the DotStar LED turns red, check the USB cable, try another USB port, etc.

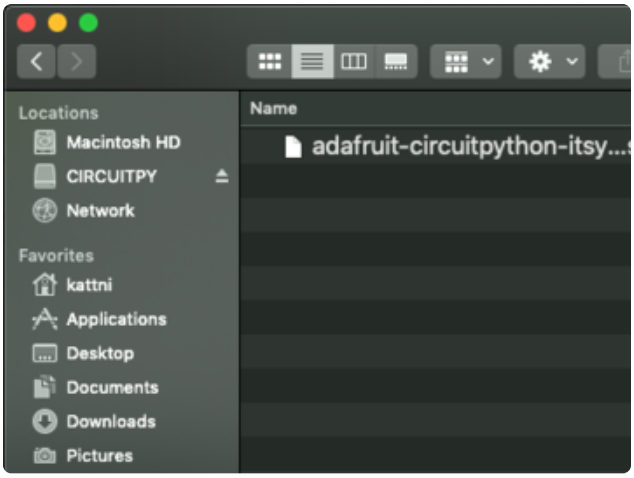
If double-clicking doesn't work the first time, try again. Sometimes it can take a few tries to get the rhythm right!



You will see a new disk drive appear called **ITSY840BOOT**.



Drag the **adafruit_circuitpython_etc.uf2** file to **ITSY840BOOT.**



The LED will flash. Then, the **ITSY840BOOT** drive will disappear and a new disk drive called **CIRCUITPY** will appear.

That's it, you're done! :)

CircuitPython code

This necklace code uses the excellent Adafruit Bluefruit app, you can get it free for both Android and iOS. Read more about what this app can do in its official guide: Bluefruit LE Connect for iOS and Android (https://adafru.it/DNc)!

iOS App

https://adafru.it/ddu

Android App

https://adafru.it/f4G

Now that you've setup CircuitPython on the ItsyBitsy and downloaded the app from the links above, we can take a look at the code! Skip towards the bottom of this page if you want to see the entire code right away. We'll go through some of the main parts of the code before then.

The main `while` loop will be structured like this:

```
# Initial empty state
state = ""

while True:
    # Advertise when not connected.
    ble.start_advertising(advertisement)

    while not ble.connected:
        # do something while ble is not yet connected

    while ble.connected:
        # Receive packets from Adafruit Bluefruit app
        # Set the state to prevent redundant hits
        # Act upon the current state
        # Also handle touch interrupt
```

The behavior when there is no BLE connection is simple: we detect touch, and then randomly choose Y or N to display. If there's no touch, we display a default, very classy, twinkling animation. Now, you can ask a question of the necklace, and the random number gods will give you the answer you seek!

```
if touch.value:
  yes_or_no() # Randomly displays 'Y' for yes and 'N' for no.
else:
  twinkle() # Keep it classy.
```

When we are connected over BLE, we then watch out for any packets that come in from the app, and we set the value of the `state` variable accordingly. Setting this state will keep behavior deterministic and will prevent any app double-taps from affecting the necklace. Note that to keep it simple, I've opted to simply use strings and if-statements to track the state here, but you could also use constant variables and/or other data structures to optimize further.

```
# Set state string based on pressed button from Bluefruit app
# and to prevent redundant hits
if isinstance(packet, ButtonPacket) and packet.pressed:
  # UP button pressed
  if packet.button == ButtonPacket.UP and state != "chase":
    state = "chase"
    # DOWN button
    elif packet.button == ButtonPacket.DOWN and state != "comet":
```

```
        state = "comet"
        # ...
```

Finally, we display animations based on the value of the state string, and we choose a different default animation just to indicate to the necklace viewer that Bluetooth is indeed connected. Touch is in a separate if-statement so that it can be triggered in the middle of the other animations.

```
# Touch is handled as a separate state
if touch.value:
    yes_or_no()

# Act upon the state
if state == "chase":
    chase.animate()
elif state == "comet":
    rainbow_comet.animate()
elif state == "rainbowchase":
    rainbow_chase.animate()
elif state == "hello":
    pixels.fill(0)
    scroll_text(packet, SCROLL_TEXT_CUSTOM_WORD)
else:
    chase.animate()
```

Installing Project Code

To use with CircuitPython, you need to first install a few libraries, into the lib folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, open the directory **ItsyBitsy_DotStar_Necklace/** and then click on the directory that matches the version of CircuitPython you're using and copy the contents of that directory to your **CIRCUITPY** drive.

Your **CIRCUITPY** drive should now look similar to the following image:

CIRCUITPY

```
# SPDX-FileCopyrightText: 2020 Anne Barela for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time
import adafruit_dotstar
import board
import random
import touchio
from adafruit_pixel_framebuf import PixelFramebuffer
from adafruit_led_animation.animation.rainbowchase import RainbowChase
from adafruit_led_animation.animation.rainbowcomet import RainbowComet
```

```
from adafruit_led_animation.animation.chase import Chase
from adafruit_led_animation.color import PINK

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

###############################################################################
# Customize variables

# Set capacitive touch pin
TOUCH_PIN = board.D11

# These are the pixels covered by the brass cap touch
# We will try to avoid using these pixels in the "twinkle" default animation
COVERED_PIXELS = [40,41,42,48,49,50,56,57,58]

# Adjust this higher if touch is too sensitive
TOUCH_THRESHOLD = 3000

# Adjust SCROLL_TEXT_COLOR_CHANGE_WAIT lower to make the color changes for
# the text scroll animation faster
SCROLL_TEXT_COLOR_CHANGE_WAIT = 5

# Change this text that will be displayed when tapping 2 on the
# Bluefruit app control pad (after connecting on your phone)
SCROLL_TEXT_CUSTOM_WORD = "hello world"

# Increase number to slow down scrolling
SCROLL_TEXT_WAIT = 0.05

# How bright each pixel in the default twinkling animation will be
TWINKLE_BRIGHTNESS = 0.1

###############################################################################
# Initialize hardware

touch_pad = TOUCH_PIN
touch = touchio.TouchIn(touch_pad)
touch.threshold = TOUCH_THRESHOLD

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

# Colors
YELLOW = (255, 150, 0)
TEAL = (0, 255, 120)
CYAN = (0, 255, 255)
PURPLE = (180, 0, 255)
TWINKLEY = (255, 255, 255)
OFF = (0, 0, 0)

# Setup Dotstar grid and pixel framebuffer for fancy animations
pixel_width = 8
pixel_height = 8
num_pixels = pixel_width * pixel_height
pixels = adafruit_dotstar.DotStar(board.A1, board.A2, num_pixels, auto_write=False,
brightness=0.1)
pixel_framebuf = PixelFramebuffer(
    pixels,
    pixel_width,
    pixel_height,
    rotation=1,
    alternating=False,
    reverse_x=True
```

```
)
# Fancy animations from https://learn.adafruit.com/circuitpython-led-animations
rainbow_chase = RainbowChase(pixels, speed=0.1, size=3, spacing=6, step=8)
chase = Chase(pixels, speed=0.1, color=CYAN, size=3, spacing=6)
rainbow_comet = RainbowComet(pixels, speed=0.1, tail_length=5, bounce=True,
colorwheel_offset=170)


def scroll_framebuf_neg_x(word, color, shift_x, shift_y):
    pixel_framebuf.fill(0)
    color_int = int('0x%02x%02x%02x' % color, 16)

    # negate x so that the word can be shown from left to right
    pixel_framebuf.text(word, -shift_x, shift_y, color_int)
    pixel_framebuf.display()
    time.sleep(SCROLL_TEXT_WAIT)

def scroll_text(packet, word):
    # scroll through entire length of string.
    # each letter is always 5 pixels wide, plus 1 space per letter
    scroll_len = (len(word) * 5) + len(word)
    color_list = [CYAN, TWINKLEY, PINK, PURPLE, YELLOW]

    color_i = 0
    color_wait_tick = 0
    # start the scroll from off the grid at -pixel_width
    for x_pos in range(-pixel_width, scroll_len):
        # detect touch
        if touch.value:
            pixel_framebuf.fill(0)
            pixel_framebuf.display()
            return;

        # detect new packet
        if isinstance(packet, ButtonPacket) and packet.pressed:
            return;

        color = color_list[color_i]
        scroll_framebuf_neg_x(word, color, x_pos, 0)

        # Only change colors after SCROLL_TEXT_COLOR_CHANGE_WAIT
        color_wait_tick = color_wait_tick + 1
        if color_wait_tick == SCROLL_TEXT_COLOR_CHANGE_WAIT:
            color_i = color_i + 1
            color_wait_tick = 0

        if color_i == len(color_list):
            color_i=0

    # wait a bit before scrolling again
    time.sleep(.5)

# Manually chosen pixels to display "Y"
# in the proper orientation
def yes(color):
    pixels[26] = color
    pixels[27] = color
    pixels[28] = color
    pixels[36] = color
    pixels[44] = color
    pixels[21] = color
    pixels.show()
    time.sleep(0.1)
    pixels.fill(0)

# Manually chosen pixels to display "N"
# in the proper orientation
def no(color):
    pixels[26] = color
```

```
    pixels[19] = color
    pixels[12] = color
    pixels[27] = color
    pixels[28] = color
    pixels[29] = color
    pixels[30] = color
    pixels[37] = color
    pixels[44] = color
    pixels.show()
    time.sleep(0.1)
    pixels.fill(0)

def yes_or_no():
    pixels.fill(0)
    print(touch.raw_value)
    value = 0
    pick=0

    pick = random.randint(0,64)
    time.sleep(0.1)

    if pick % 2:
        print('picked yes!');
        yes(PINK)
        time.sleep(1)
    else:
        print('picked no!');
        no(TEAL)
        time.sleep(1)


def twinkle_show():
    pixels.brightness = TWINKLE_BRIGHTNESS
    pixels.show()
    time.sleep(.1)
    if touch.value:
        return;

def twinkle():
    # randomly choose 3 pixels
    spark1 = random.randint(0, num_pixels-1)
    spark2 = random.randint(0, num_pixels-1)
    spark3 = random.randint(0, num_pixels-1)

    # make sure that none of the chosen pixels are covered
    while spark1 in COVERED_PIXELS:
        spark1 = random.randint(0, num_pixels-1)
    while spark2 in COVERED_PIXELS:
        spark2 = random.randint(0, num_pixels-1)
    while spark3 in COVERED_PIXELS:
        spark3 = random.randint(0, num_pixels-1)

    # Control when chosen pixels turn on for dazzling effect
    pixels[spark1] = TWINKLEY
    pixels[spark2] = OFF
    pixels[spark3] = OFF
    twinkle_show()
    pixels[spark1] = TWINKLEY
    pixels[spark2] = TWINKLEY
    pixels[spark3] = OFF
    twinkle_show()
    pixels[spark1] = TWINKLEY
    pixels[spark2] = TWINKLEY
    pixels[spark3] = TWINKLEY
    twinkle_show()
    pixels[spark1] = OFF
    pixels[spark2] = TWINKLEY
    pixels[spark3] = TWINKLEY
    twinkle_show()
```

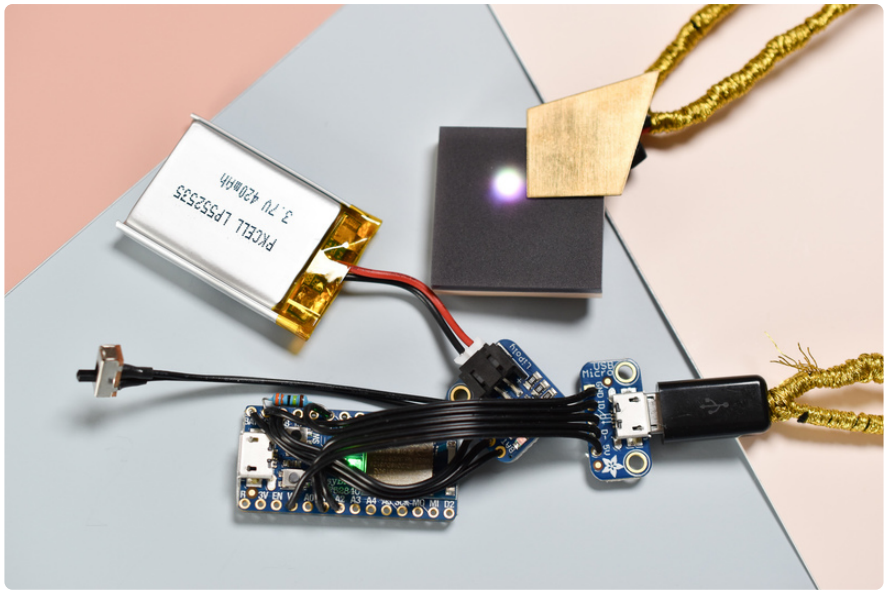
```
    pixels[spark1] = OFF
    pixels[spark2] = OFF
    pixels[spark3] = TWINKLEY
    twinkle_show()

    pixels.fill(OFF)
    pixels.show()
    time.sleep(0.6)

# Initial empty state
state = ""

while True:
    # Advertise when not connected.
    ble.start_advertising(advertisement)

    while not ble.connected:
        if touch.value:
            yes_or_no()
        else:
            twinkle()

    while ble.connected:
        # Set the state
        if uart_service.in_waiting:
            # Packet is arriving.
            packet = Packet.from_stream(uart_service)

            # set state string based on pressed button from Bluefruit app
            # and to prevent redundant hits
            if isinstance(packet, ButtonPacket) and packet.pressed:
                # UP button pressed
                if packet.button == ButtonPacket.UP and state != "chase":
                    state = "chase"
                # DOWN button
                elif packet.button == ButtonPacket.DOWN and state != "comet":
                    state = "comet"
                # 1 button
                elif packet.button == '1' and state != "rainbowchase":
                    state = "rainbowchase"
                # 2 button
                elif packet.button == '2' and state != "hello":
                    state = "hello"

        # Touch is handled as an interrupt state
        if touch.value:
            yes_or_no()

        # Act upon the state
        if state == "chase":
            chase.animate()
        elif state == "comet":
            rainbow_comet.animate()
        elif state == "rainbowchase":
            rainbow_chase.animate()
        elif state == "hello":
            pixels.fill(0)
            scroll_text(packet, SCROLL_TEXT_CUSTOM_WORD)
        else:
            chase.animate()
```

The Dotstar Matrix can get quite hot when all or most of the LEDs are on, so try to make sure you're testing any new animations on a table and checking how hot

it gets first before wearing the necklace. However, the animations included above shouldn't result in too much heat.

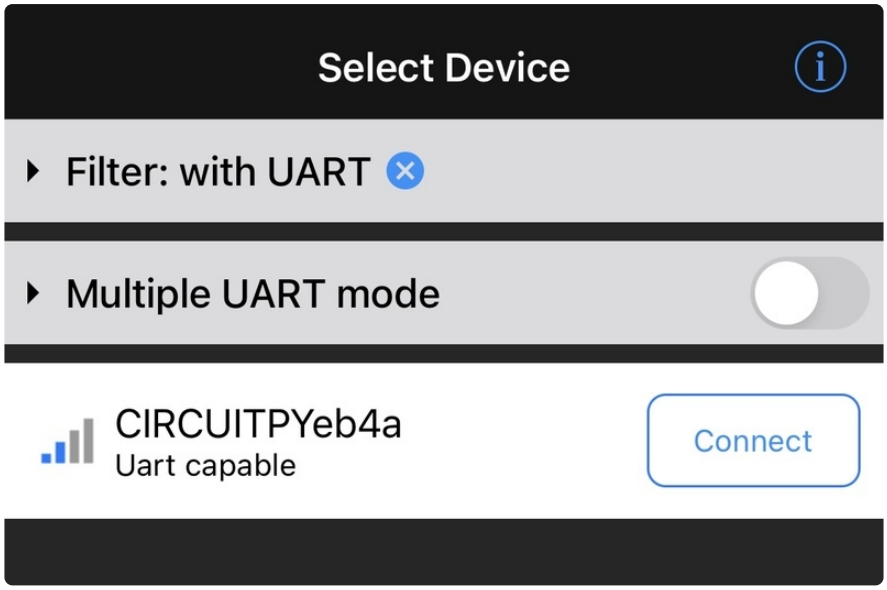
Test capacitive touch

Now you can connect the necklace via the microB port to test that the twinkling animation and capacitive touch sensing is working. You may need to adjust the threshold of the capacitive touch if it's too sensitive or not sensitive enough.



Test Bluefruit app interaction

You can now test the interaction with the necklace using the Bluefruit app. Turn on the power switch, and open the Bluefruit app -- you should see your device like so:



Then, press "Connect" and then "Controller" and then "Control Pad". You can then click on the following buttons in the Control Pad screen and see the animations change in the necklace:

- Up button
- Down button
- "1" button
- "2" buton

Go ahead and change the corresponding animations in the code to whatever you choose!

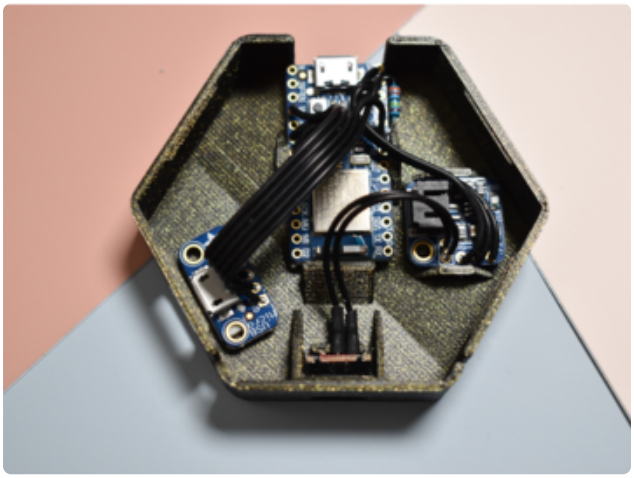
We're almost done. Head on over to final assembly to stuff everything into an enclosure!

Final Assembly

You can either choose the 3d printed case or the laser cut case. I recommend the 3d printed case since it's a very slight mod of the excellent case from the Neopixel LED Heart Necklace tutorial (https://adafru.it/Pod), and it allows access to the reset button.

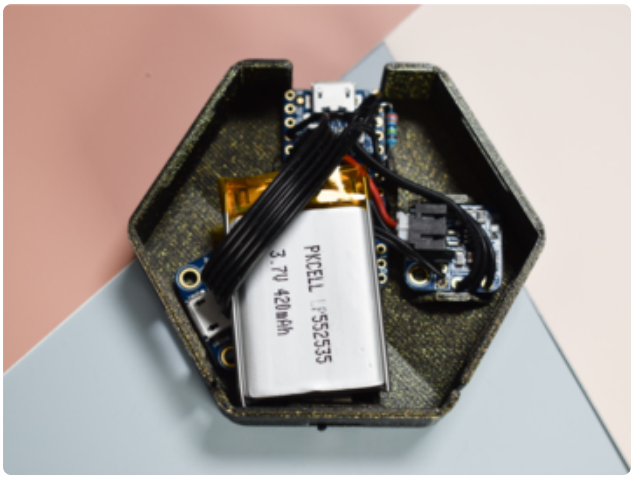
If you want something a bit smaller and compact though, the laser cut case is provided here as well (it's slimmer in width, but thicker in depth).

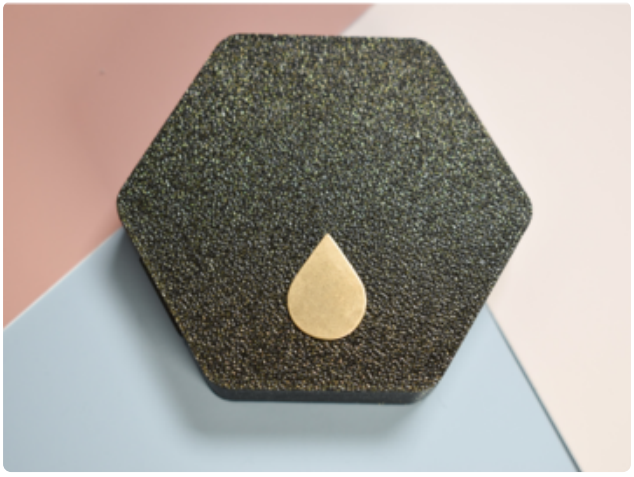
3D printed hex case



Carefully slot the ItsyBitsy, LiPo backpack and slide switch as shown -- the model will have dedicated structures to support these parts.

Align the microB usb port to the remaining hole to the left of the switch. Secure it using some double sided foam tape, and make sure that it's right up against the edge of the case. Test that the necklace's microB plug will fit into the plug through the case hole and adjust accordingly.



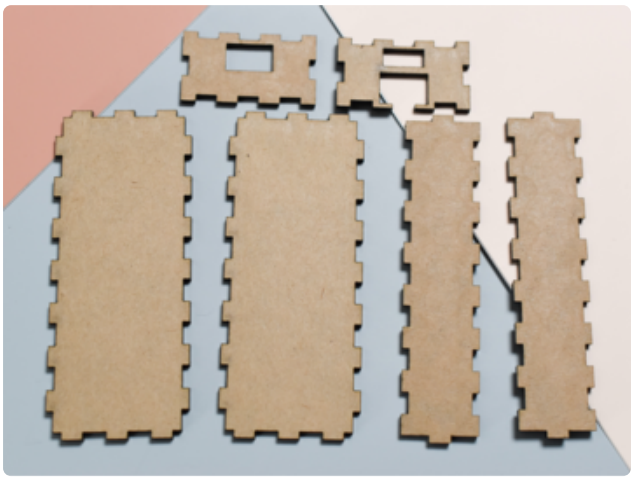


Connect the battery to the LiPo backpack and place on top of the ItsyBitsy as shown. You may want to tuck it under the wires. There should be no tape needed to secure this battery.

Nudge the wires away from the RESET button before putting on the case cover, making sure to align the RESET button with the button extender.

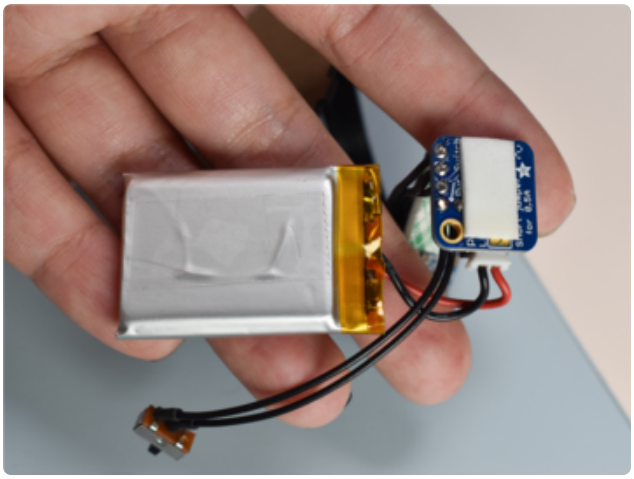
Optionally, add another brass shape to this case for extra bling.

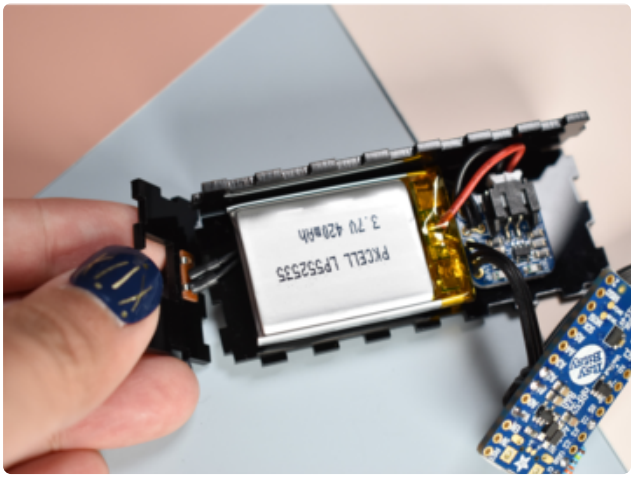
Laser cut rectangular case

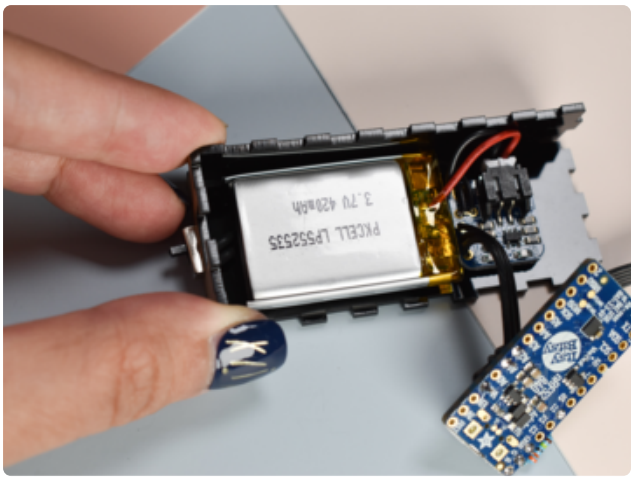


This case is made up of 4 different types of panels, with finger joints that slot into each other:

1. Top/bottom panels
2. Side panels
3. Necklace port panel
4. Power and charging port panel





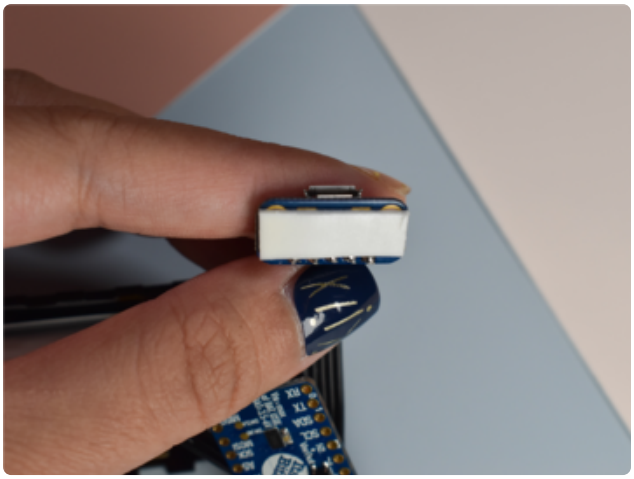


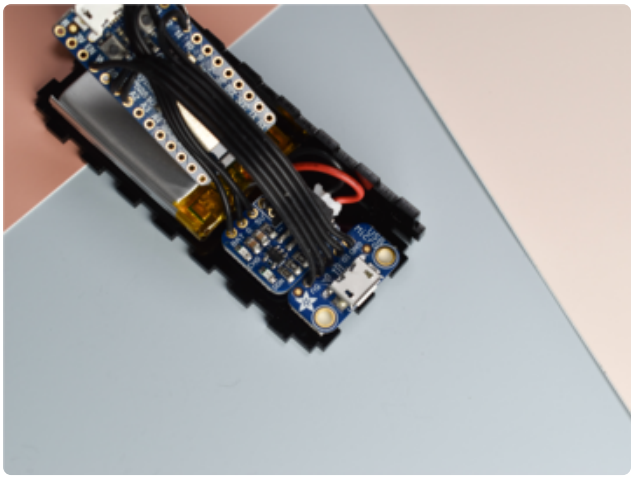
Add thin tape to LiPo battery and foam tape to LiPo backpack

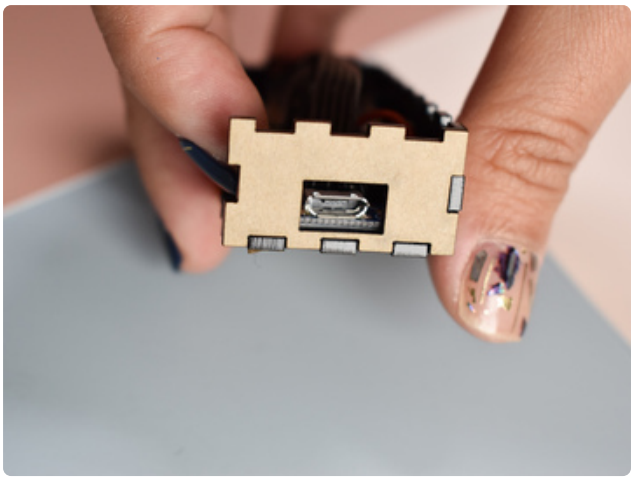
Then, with the bottom panel and one of the side panels attached perpendicularly, place the battery and the LiPo backpack to the bottom panel as shown in the photo.

The thin tape is needed for the battery so that the ItsyBitsy can fit on top of it and align with Panel #4 (as referenced above).

Fit the switch in the appropriate cutout in Panel #4 and fit that panel with the rest of the case.



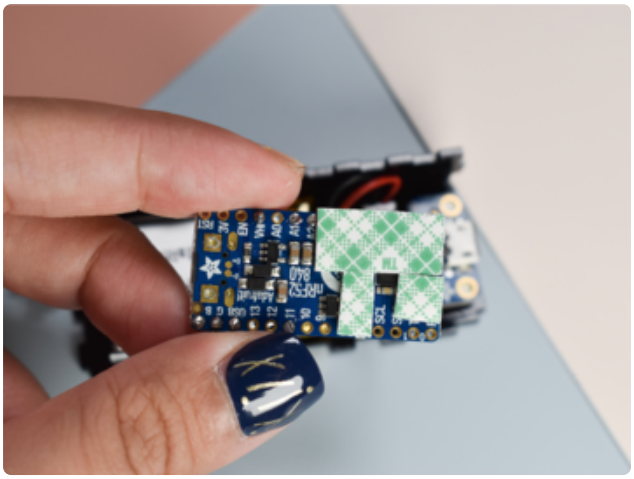


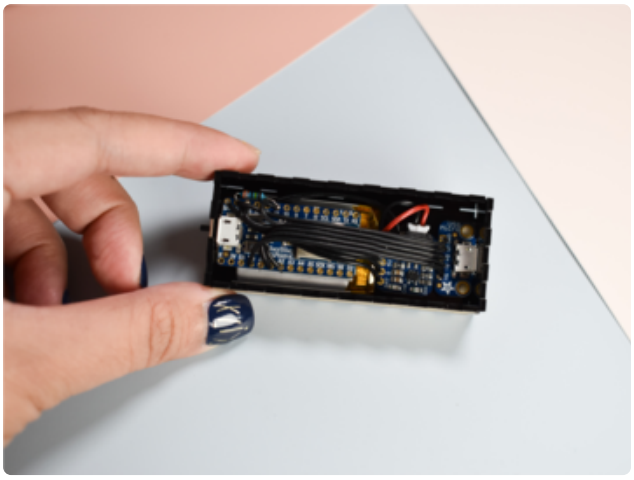


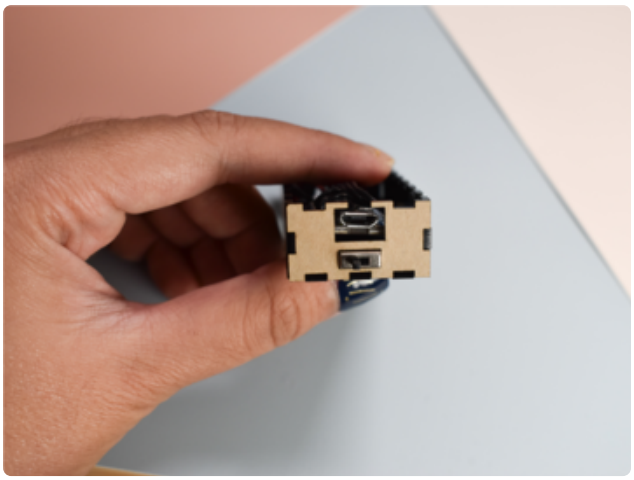
Secure the USB MicroB port in place

Use foam tape to secure the microB port as shown in the photo. The edge of the port should be as close as possible to the inner edges of the finger joints.

Then, attach Panel #3 (necklace port panel), and make sure that the microB port is aligned with the cutout.







Place the ItsyBitsy on top of the battery

Cut some foam tape to fit in the flat areas on the bottom of the board. The foam tape should be thick enough to allow the ItsyBitsy to lay flat. The amount of tape shown in the photo should provide enough resistance when inserting a micro USB plug into the board.

Secure the ItsyBitsy on top of the battery and make sure to align the edge of the ItsyBitsy right up against the panel, centering the microUSB port with the cutout on the panel.

Close up the case

Once you're happy with how everything is laid out, you can use super glue or any other appropriate glue on the finger joints to solidify the case.

Secure and glue the last top panel, and if desired, add a bit of flair with another brass shape. The contrast of brass and black here is simply beautiful!

And that's it! You can now light up the night (or day) with your very own modern geometric LED matrix necklace. Ask the necklace a question, touch for the answer, and may the Fortunes smile upon you.

Below is a photo of the combination of the 3D printed hex case and the laser cut pendant frame, which is definitely my fave combo. Since the parts of this project is quite modular, there's lots of room for customization, and with CircuitPython it's quite easy to change the LED animations. Have fun modifying this project so that it suits your own style!