adafruit learning system

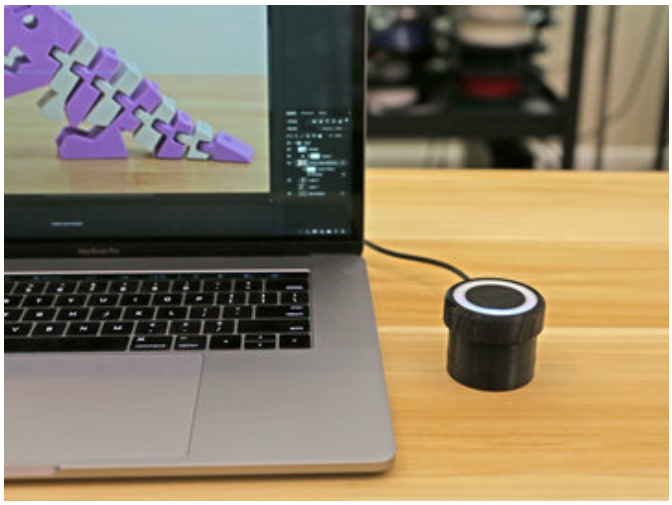
CircuitPython Media Dial

Created by Ruiz Brothers

Last updated on 2018-08-22 04:05:39 PM UTC

Guide Contents

Overview



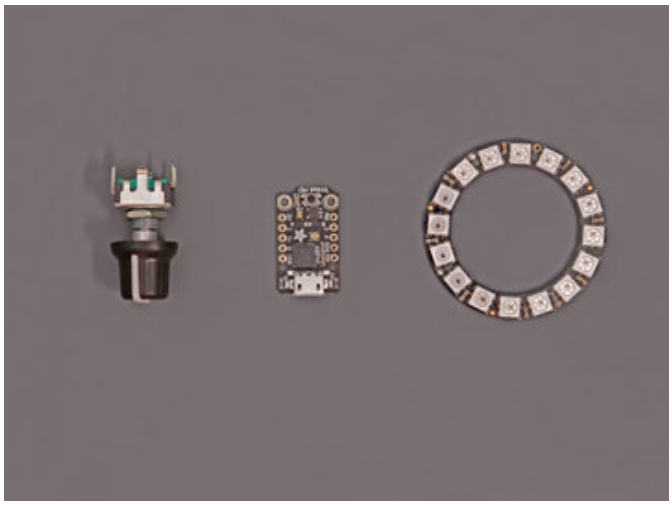
This is a DIY multimedia dial. It rotates like a knob and clicks like a mouse.

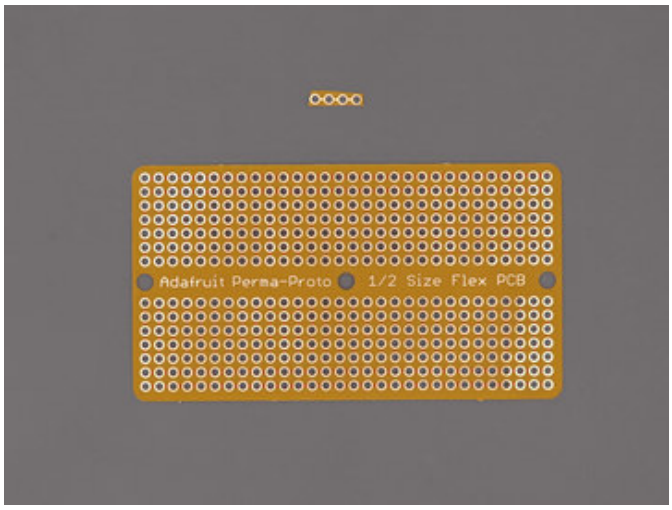
It works well as an assignable USB controller that you can program to do just about anything you want.

It's powered by Adafruit's Circuit Python so you can quickly program it to run key commands in any application.

So you can use it to edit videos, control youtube or even scroll through long documents.

Inside the 3D printed knob is a rotary encoder and NeoPixel ring. As you turn the knob, an LED follows the direction making it easy to see where it is.





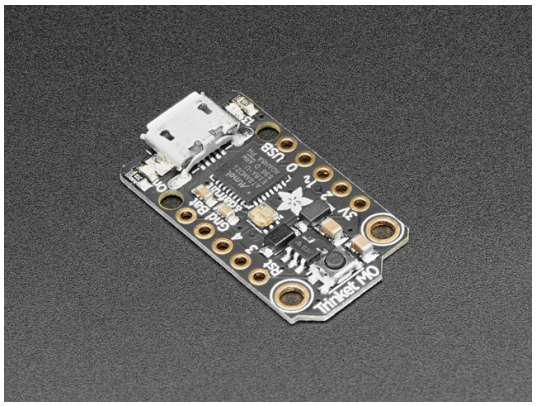
This uses the Adafruit HID Library for Circuit Python. All of the key codes are listed in the docs so it's easy to customize your own key commands.

It's powered by Adafruit's Trinket M0 and runs Circuit Python.

Prerequisite Guides

If your new to electronics and *CircuitPython*, I suggest you walk through the following guides to get the basics. The Adafruit Feather M0 Express guide will walk you through setting it up with CircuitPython. See the DotStar guide for more information on how they work.

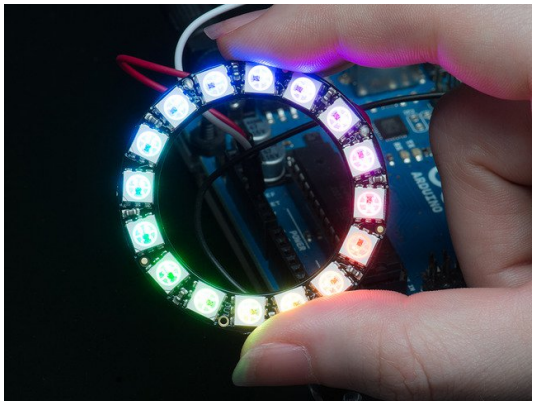
- Collin's Lab – Soldering (https://adafru.it/wsa)
- Welcome to CircuitPython! (https://adafru.it/Bfx)



Adafruit Trinket M0 - for use with CircuitPython & Arduino IDE

$8.95
IN STOCK

ADD TO CART



NeoPixel Ring - 16 x 5050 RGB LED with Integrated Drivers

$9.95
IN STOCK

ADD TO CART

Filament for 3D Printers in Various Colors and Types

$0.00
OUT OF STOCK

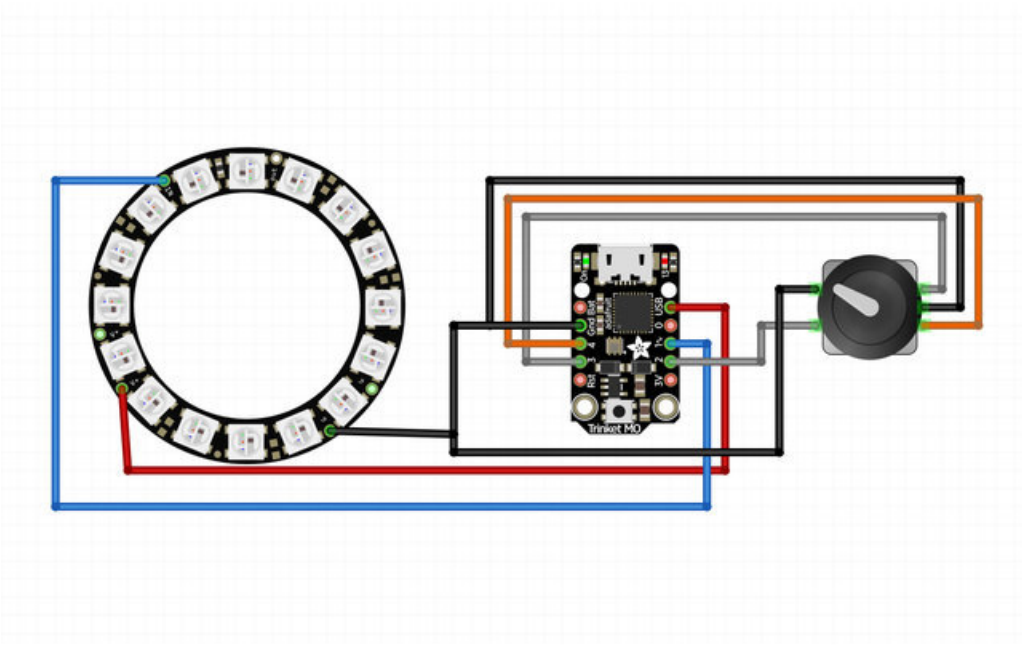
OUT OF STOCK

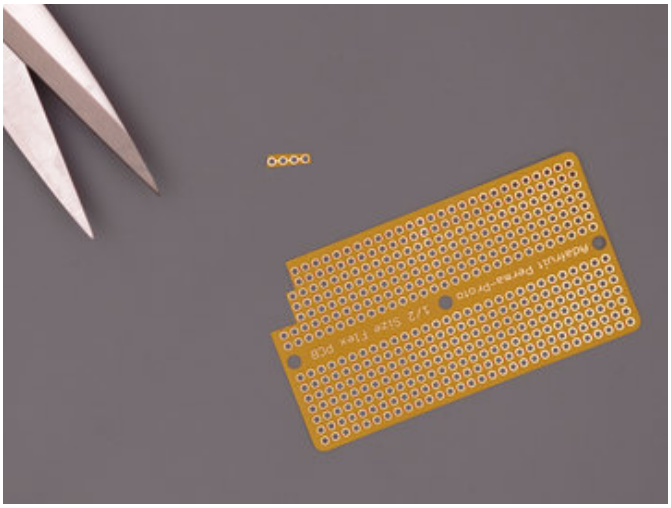
Ultimaker 3 - 3D Printer

$3,750.00
IN STOCK

ADD TO CART

Circuit Diagram





Trinket Ground Extension

The Trinket only has one ground pin. Add more grounds with the help of a flex perma-proto.

Cut a piece with four connections on a continuous rail to fit the three ground connections need for each component.

Take a moment to review the components in the circuit diagram. This illustration is meant for referencing wired connections - The length of wire, position and size of components are not exact.

The **NeoPixel Ring** will connect to Trinket M0. Measure **90mm** of wire for power, ground and data in.
Data IN connects to pin **1** on the Trinket
Ground connects to **GND** on the Trinket
5v connects to **USB** on the Trinket

The **Rotary Encoder** connects to the Trinket with **80mm long wires** for all of it's connections.

Three pin side:

Middle connects to ground on the Trinket
Top pin connects to **pin 3** on the Trinket
Bottom pin connects to **pin 4** on the Trinket

Two pin side:

Top pin connects to **GND** on the Trinket
Bottom pin connects to **pin 2** on the Trinket

Code

Setup Adafruit Trinket M0 for CircuitPython

We'll need to get our board setup so we can run CircuitPython code. First thing we'll need to do is connect the board to your computer with a microUSB cable. Then double-click on the reset button to put it in "UF2" boot-loader mode. The NeoPixel will turn green. The board will then show up as a USB storage device on your computer named "TRINKETBOOT".

Follow the guide below to setup the firmware, once complete, come back here and proceed.

https://adafru.it/ABS

https://adafru.it/ABS

Download Adafruit CircuitPython Library Bundle

In order to run the code, we'll need to download some libraries. The download linked below will contain all the libraries available for Circuit Python. To run the code for this project, we only need a few. Unzip the downloaded file and look for the following libraries.

Required Libraries

- Adafruit Neopixel – neopixel.mpy
- Adafruit HID – adafruit_hid

https://adafru.it/ABT

https://adafru.it/ABT

Install Circuit Python Libraries

Now that we have all of the libraries and know which ones this project needs, we'll need to copy them onto the Trinket M0 USB drive (which will be named CIRCUITPY after flashing the firmware). In the CIRCUITPY drive, create a new folder and name it "lib". Then, copy the libraries to that "lib" folder. The lib folder should contain **neopixel.mpy** and **adafruit_hid** .

Upload Code

OK, now it's time to upload the code for this project onto the CIRCUITPY drive. Create a new text document using a text app. Then, copy the code below and paste it into that newly created text document. Save that text document to the CIRCUITPY drive and name it "**main.py**". Once saved, the code will automatically run and will start working.

Modify Key Codes

You can customize the key codes to form custom commands which can be multiple keys or have it execute just single keyboard characters.

The rotary encode can execute up to 3 different commands. Pressing the knob and turning the knob left or right. These are commented in the code and can be changed by adjusting the key code value.

List of USB HID Keycodes

The long list of available keyboard characters are listed in the webpage linked below. Most of the characters are for USA keyboard only. Function keys and modifiers can be used but only some special characters are not supported.

https://adafru.it/AOi

https://adafru.it/AOi

Starting with the first command, turning the knob to the right, will execute the ctrl+up arrow keys. These are two different keyboard characters that are separated with commas. This will essentially press the two keys simultaneously. The values inside the parentheses kbd.press(**keycode.THISKEY**) are the ones you want to change. For example, the block of code below is executed when turning the knob to the right.

```
# Check if rotary encoder went up
if encoder_direction == 1:
kbd.press(Keycode.CONTROL, Keycode.UP_ARROW)
kbd.release_all()
```

For more information and trouble shooting, please check out the Circuit Python library guide, linked below.

https://adafru.it/ABU

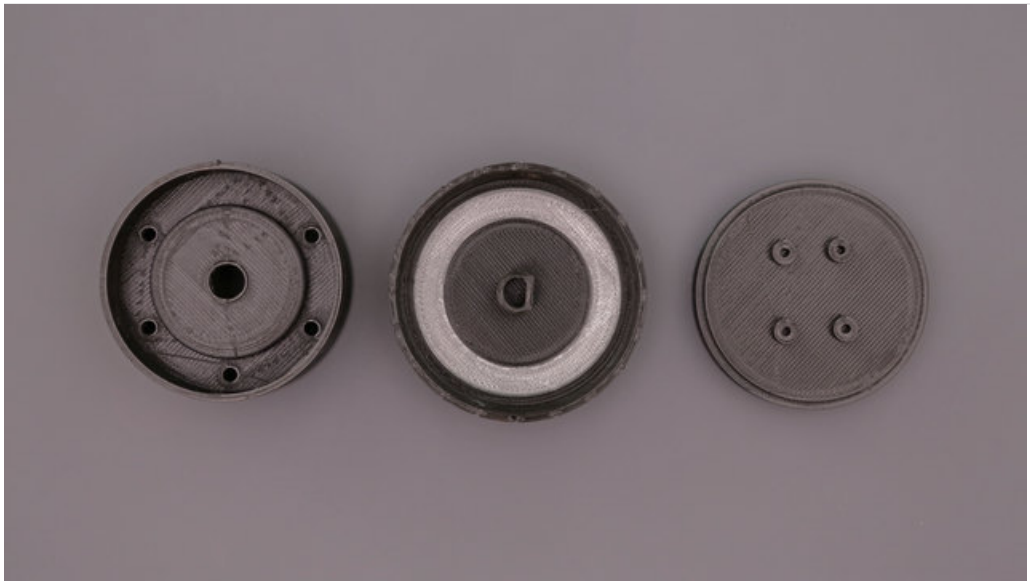
https://adafru.it/ABU

```
    """
A CircuitPython 'multimedia' dial demo
Uses a Trinket M0 + Rotary Encoder -> HID keyboard out with neopixel ring
"""

import time
from digitalio import *
from board import *
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keycode import Keycode
import neopixel

DOT_COLOR = 0xFF0000              # set to your favorite webhex color
PRESSED_DOT_COLOR = 0x008080      # set to your second-favorite color
LIT_TIMEOUT = 15                  # after n seconds, turn off ring

# NeoPixel LED ring on pin D1
# Ring code will auto-adjust if not 16 so change to any value!
ring = neopixel.NeoPixel(D1, 16, brightness=0.2)
dot_location = 0  # what dot is currently lit

# Encoder button is a digital input with pullup on D2
button = DigitalInOut(D2)
button.direction = Direction.INPUT
button.pull = Pull.UP

# Rotary encoder inputs with pullup on D3 & D4
rot_a = DigitalInOut(D3)
rot_a.direction = Direction.INPUT
rot_a.pull = Pull.UP
rot_b = DigitalInOut(D4)
rot_b.direction = Direction.INPUT
rot_b.pull = Pull.UP
```

```
# Used to do HID output, see below
kbd = Keyboard()

# time keeper, so we know when to turn off the LED
timestamp = time.monotonic()

######################### MAIN LOOP ##############################

# the counter counts up and down, it can roll over! 16-bit value
encoder_counter = 0
# direction tells you the last tick which way it went
encoder_direction = 0

# constants to help us track what edge is what
A_POSITION = 0
B_POSITION = 1
UNKNOWN_POSITION = -1  # initial state so we know if something went wrong

rising_edge = falling_edge = UNKNOWN_POSITION

# get initial/prev state and store at beginning
last_button = button.value
rotary_prev_state = [rot_a.value, rot_b.value]

while True:
    # reset encoder and wait for the next turn
    encoder_direction = 0

    # take a 'snapshot' of the rotary encoder state at this time
    rotary_curr_state = [rot_a.value, rot_b.value]

    if rotary_curr_state != rotary_prev_state:
        #print("Changed")
        if rotary_prev_state == [True, True]:
            # we caught the first falling edge!
            if not rotary_curr_state[A_POSITION]:
                #print("Falling A")
                falling_edge = A_POSITION
            elif not rotary_curr_state[B_POSITION]:
                #print("Falling B")
                falling_edge = B_POSITION
            else:
                # uhh something went deeply wrong, lets start over
                continue

        if rotary_curr_state == [True, True]:
            # Ok we hit the final rising edge
            if not rotary_prev_state[B_POSITION]:
                rising_edge = B_POSITION
                # print("Rising B")
            elif not rotary_prev_state[A_POSITION]:
                rising_edge = A_POSITION
                # print("Rising A")
            else:
                # uhh something went deeply wrong, lets start over
                continue

            # check first and last edge
            if (rising_edge == A_POSITION) and (falling_edge == B_POSITION):
```

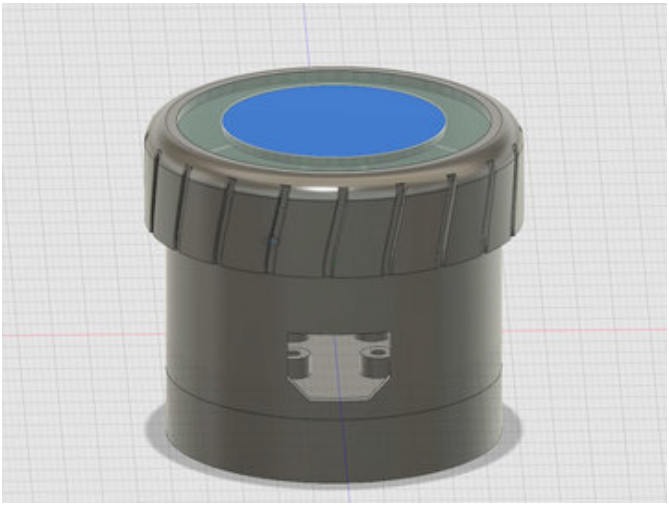
```
        if (rising_edge == A_POSITION) and (falling_edge == B_POSITION):
            encoder_counter -= 1
            encoder_direction = -1
            print("%d dec" % encoder_counter)
        elif (rising_edge == B_POSITION) and (falling_edge == A_POSITION):
            encoder_counter += 1
            encoder_direction = 1
            print("%d inc" % encoder_counter)
        else:
            # (shrug) something didn't work out, oh well!
            encoder_direction = 0

        # reset our edge tracking
        rising_edge = falling_edge = UNKNOWN_POSITION

rotary_prev_state = rotary_curr_state

# Check if rotary encoder went up
if encoder_direction == 1:
    kbd.press(Keycode.CONTROL, Keycode.UP_ARROW)
    kbd.release_all()

# Check if rotary encoder went down
if encoder_direction == -1:
    kbd.press(Keycode.CONTROL, Keycode.DOWN_ARROW)
    kbd.release_all()

# Button was 'just pressed'
if (not button.value) and last_button:
    print("Button pressed!")
    kbd.press(44) #Keycode.SPACE
    kbd.release_all()
    ring[dot_location] = PRESSED_DOT_COLOR # show it was pressed on ring
    timestamp = time.monotonic()        # something happened!
elif button.value and (not last_button):
    print("Button Released!")
    # kbd.press(Keycode.SHIFT, Keycode.SIX)
    # kbd.release_all()
    ring[dot_location] = DOT_COLOR      # show it was released on ring
    timestamp = time.monotonic()        # something happened!
last_button = button.value

if encoder_direction != 0:
    timestamp = time.monotonic()        # something happened!
    # spin neopixel LED around!
    previous_location = dot_location
    dot_location += encoder_direction   # move dot in the direction
    dot_location += len(ring)           # in case we moved negative, wrap around
    dot_location %= len(ring)
    if button.value:
        ring[dot_location] = DOT_COLOR  # turn on new dot
    else:
        ring[dot_location] = PRESSED_DOT_COLOR # turn on new dot
    ring[previous_location] = 0         # turn off previous dot

if time.monotonic() > timestamp + LIT_TIMEOUT:
    ring[dot_location] = 0   # turn off ring light temporarily
```

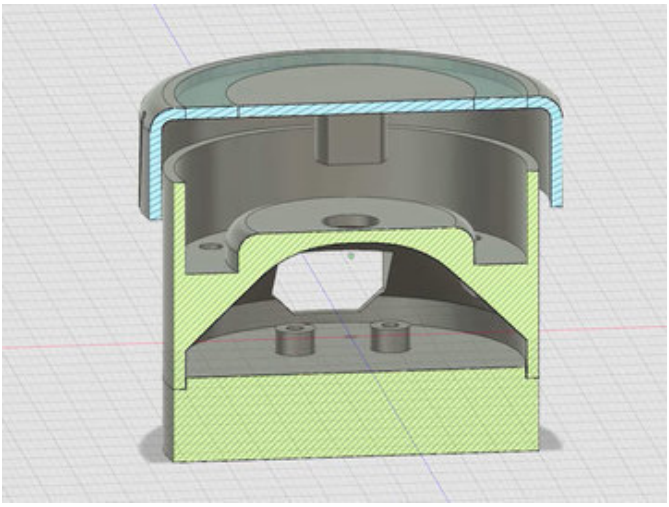
3D Printing



The 3D printed parts are fairly easy to make with most common home desktop 3D printers that are on the market.

And if you don't have access a 3D printer, you can order our parts by visiting our Thingiverse page and have someone local 3D print the parts and ship them to you.





We designed the enclosure in Autodesk Fusion 360. The case houses the electronics while the top and bottom parts will snap fit together.

We split the cover into individual pieces for 3D printing with a Dual Extruder.

The inner ring is printed in translucent material so the NeoPixel LEDs can shine through the top. To 3D print the enclosure, we used an Ultimaker 3 and slice the parts in Ultimakers CURA.

https://adafru.it/ABI

https://adafru.it/ABI

https://adafru.it/ABY

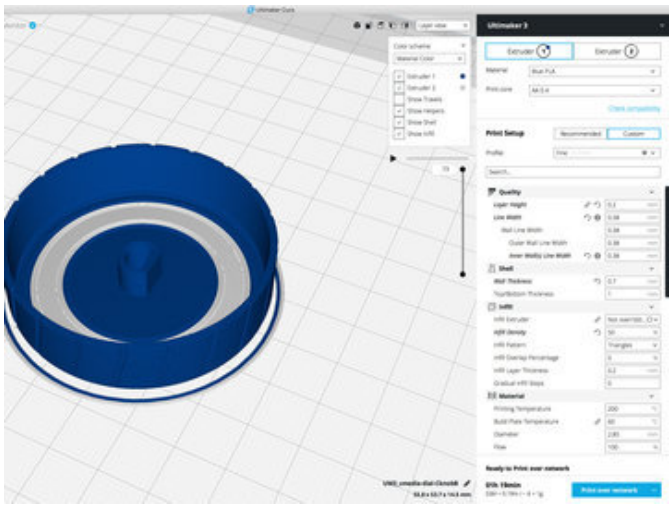
https://adafru.it/ABY

https://adafru.it/ABZ

https://adafru.it/ABZ

https://adafru.it/svF

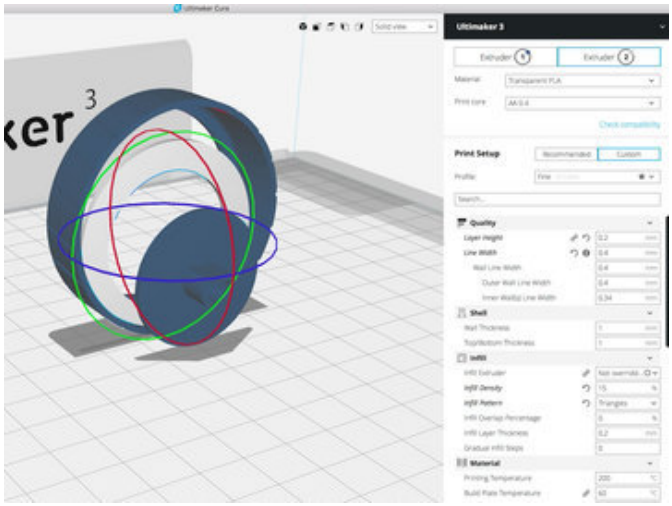
https://adafru.it/svF



Slice Settings

Download the STL files and import them into your 3D printing slicing software. You'll need to adjust your settings accordingly if you're using material different than PLA.

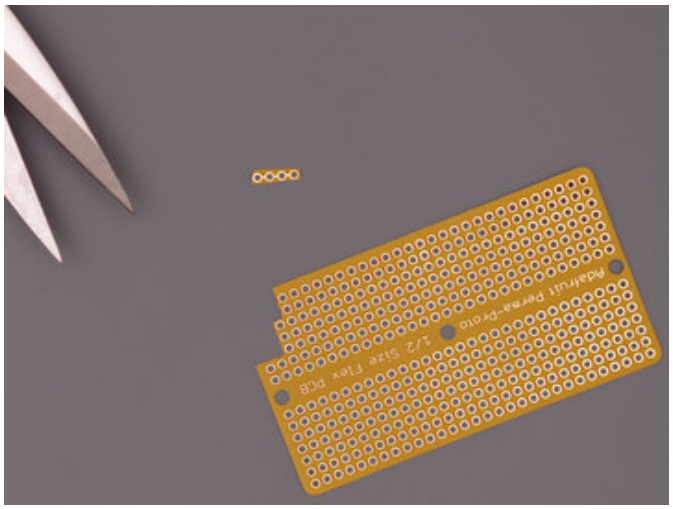
- 210C Extruder Temp
- 65c heated bed
- 1.0 Extrusion Multiplier
- .4mm Nozzle
- 0.38 Extrusion Width
- .2mm Layer Height
- 30% infill
- No Supports
- skirt
- 60mm/s | 120mm travel speed



Dual Colors

We included a dual color version of the knob cover to allow the NeoPixel ring to diffuse it. Use your preferred slicer to align and combine all three parts that make up the individual parts of the knob. Use transparent filament to allow the led lights to diffuse the knob cover.

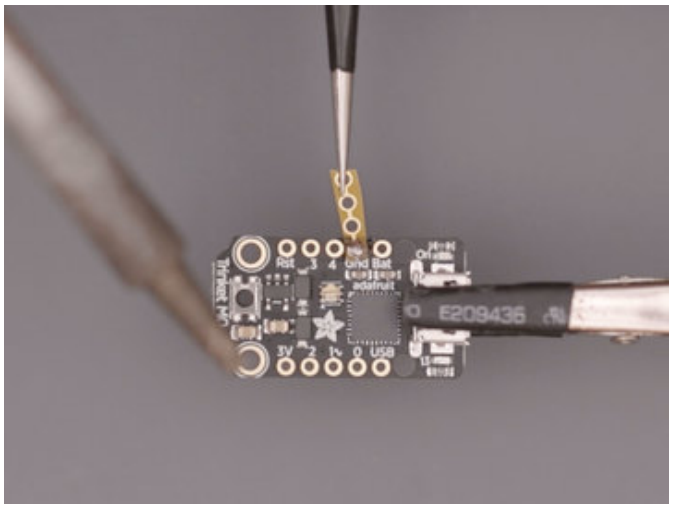
Assemble



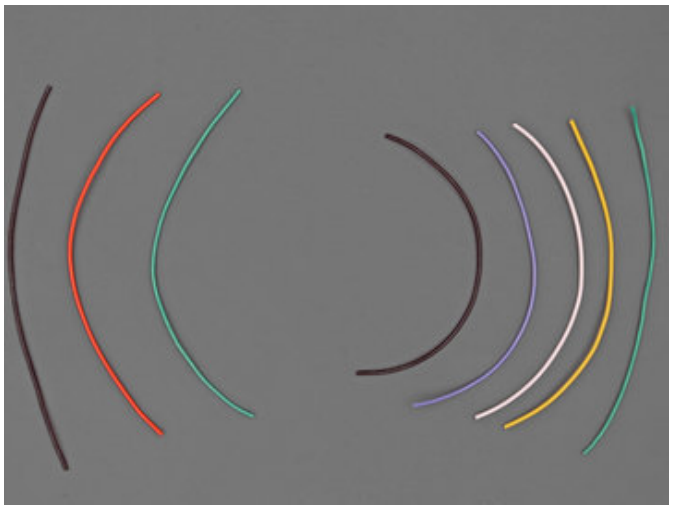
Prep Trinket M0

To assemble the circuit we used a flexible PCB to extend the ground connections.This way we can break out more pins on the Trinket M0.

Use sharp scissors to cut a small piece of four connected pins. The first pin on the flex PCB is soldered to the GND pin on the Trinket. The other three pins will be used for the NeoPixel Ring and Rotary encoder.



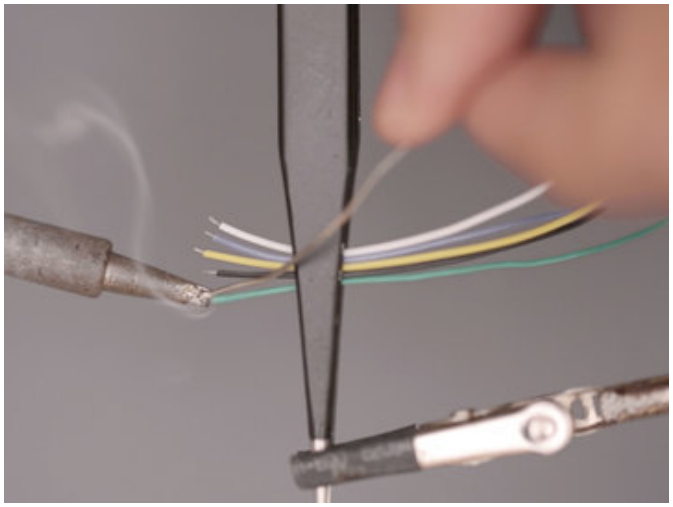
Tin the GND pin on the Trinket and then using tweezers, hold the flex PCB over the GND pin while apply heated to solder the Flex PCB to the Trinket.

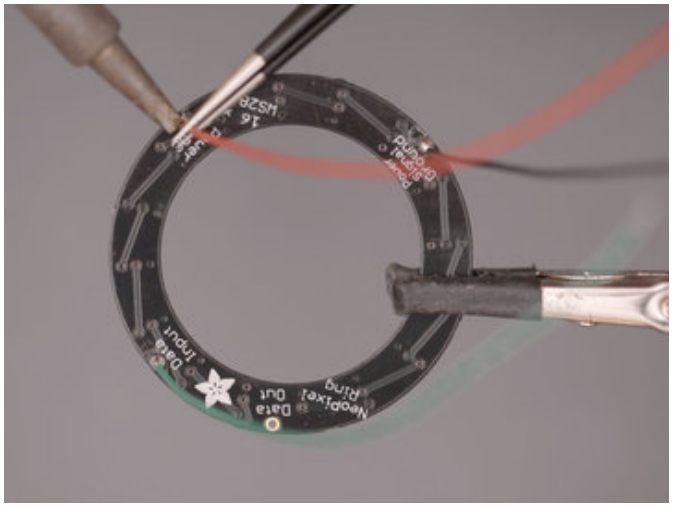


Prep wires

Next we'll move on to measuring, cutting and tinning all of the wires needed to connect all of the components together.

The **NeoPixel Ring** will need all of the connections to be **90mm** long. The R**otary Encoder** wires need to be **80mm** long to reach the Trinket.



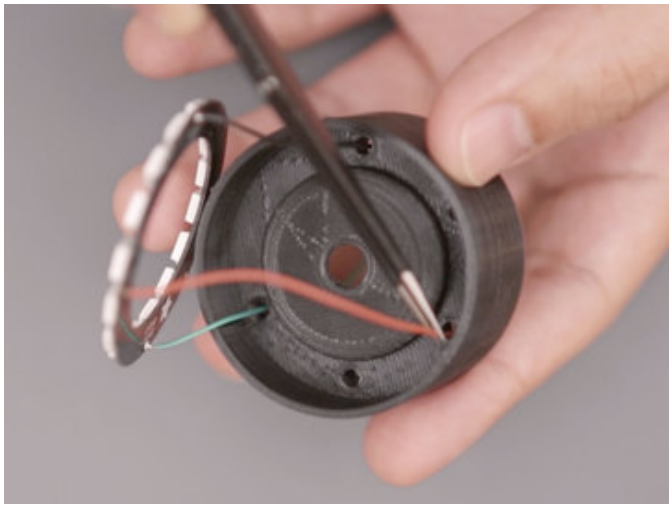


Solder NeoPixel Ring

Use third helping hands to hold the NeoPixel ring while tinning and soldering wires to the Data IN, 5v and Ground pins.

Thread Ring wires

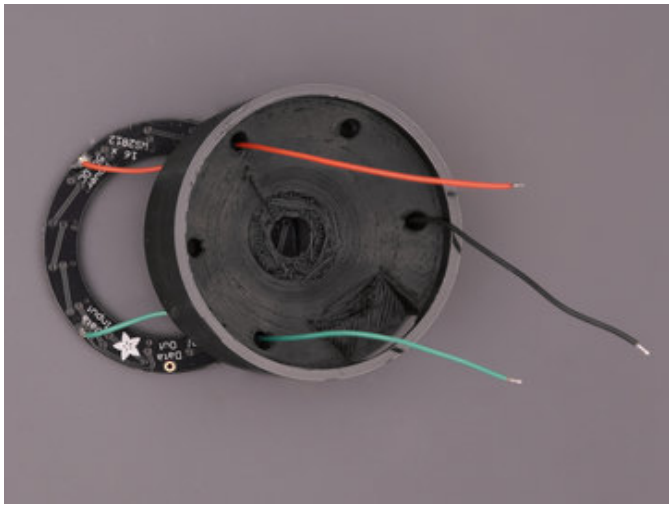
Align the NeoPixel ring over the mounting area on the **dial-mid** part and pass each wire through the holes on the part.

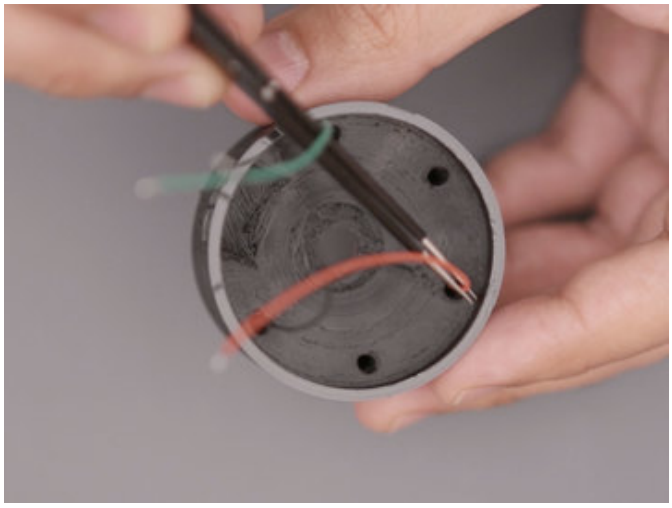


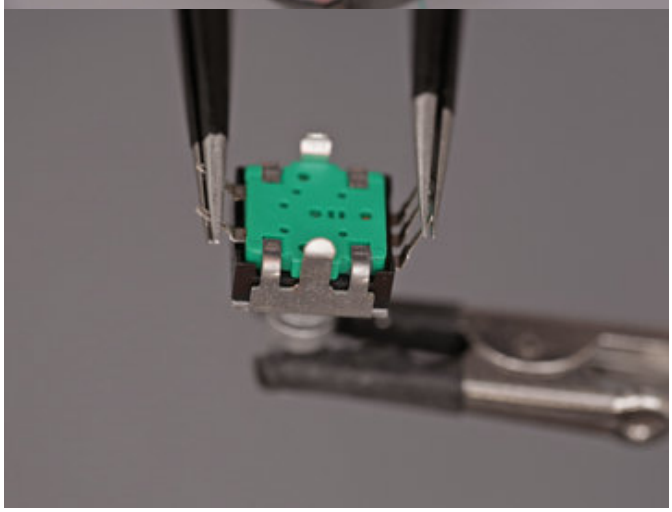
Use tweezers to help thread the wires through the holes.

Gently press down on the NeoPixel Ring to snap fit inside the mount on the enclosure.

You can use the other two free holes to push the ring out of the enclosure if you ever need to.





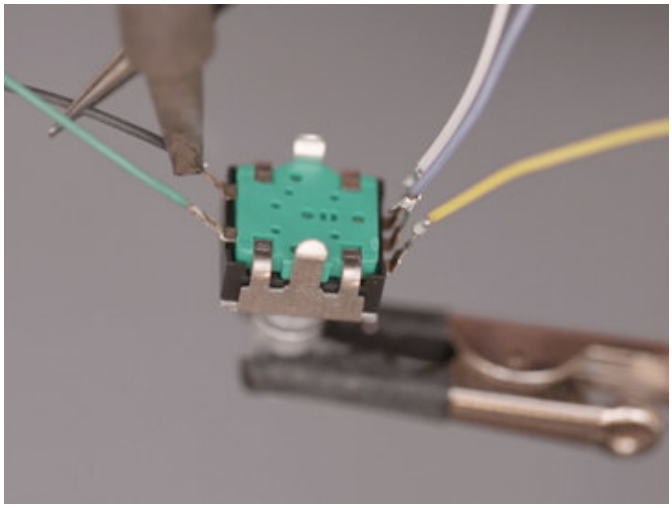


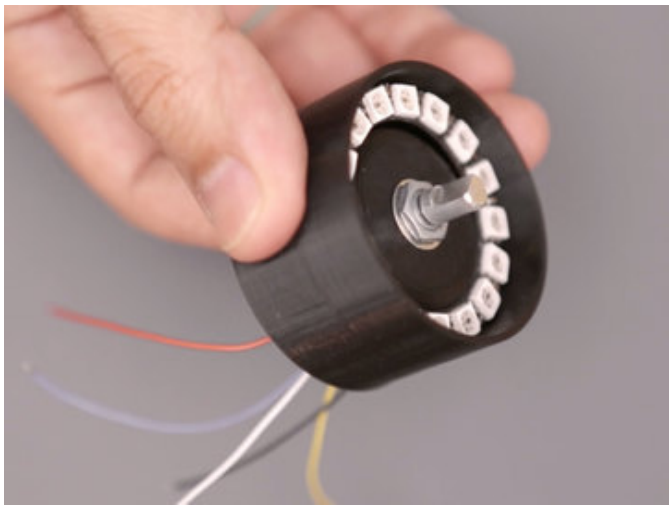
Solder Rotary

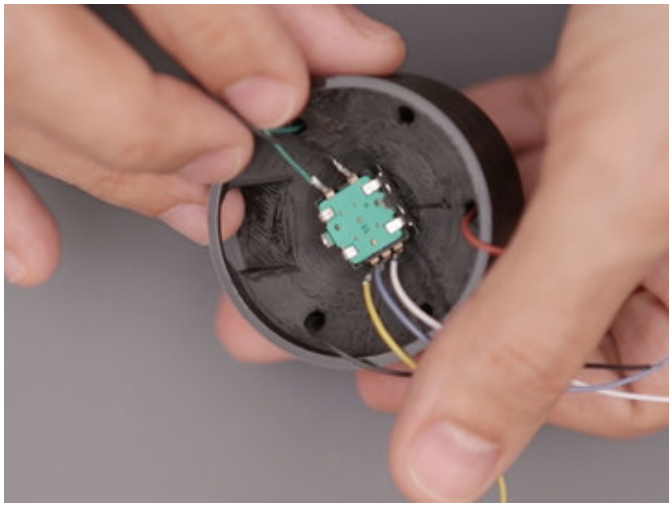
Now we can prepare the Rotary Encoder. First, gently bend the legs back to avoid making contact with the Trinket once mounted inside the enclosure.

Tin and then solder each colored coded wire to according to the circuit diagram.

You can hold the Rotary Encoder by the steam with a third helping hands to make it easier to solder.



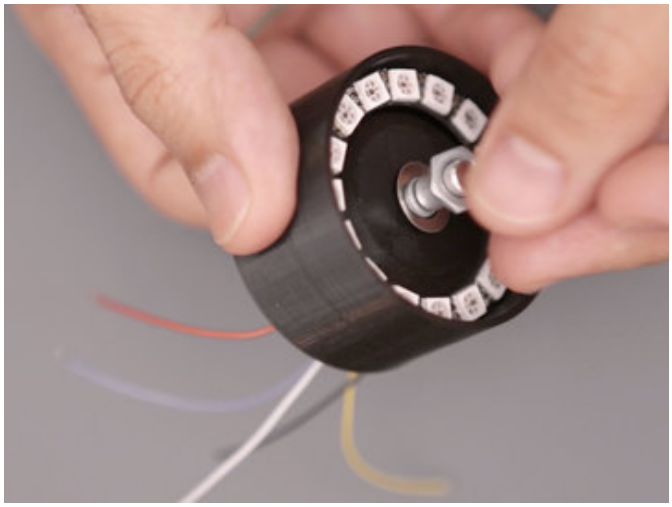




Mount Rotary Encoder

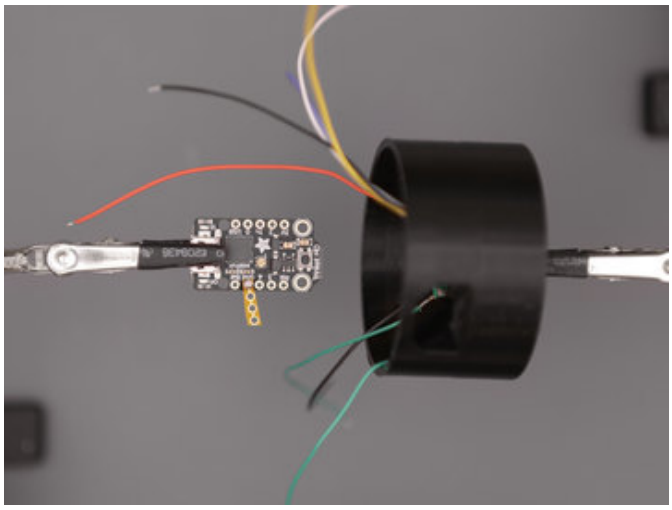
The rotary encoder is fitted through the hole on the **dial-mid** part. Rotate the part until the legs are aligned with the USB port on the side of the enclosure as shown in the picture.

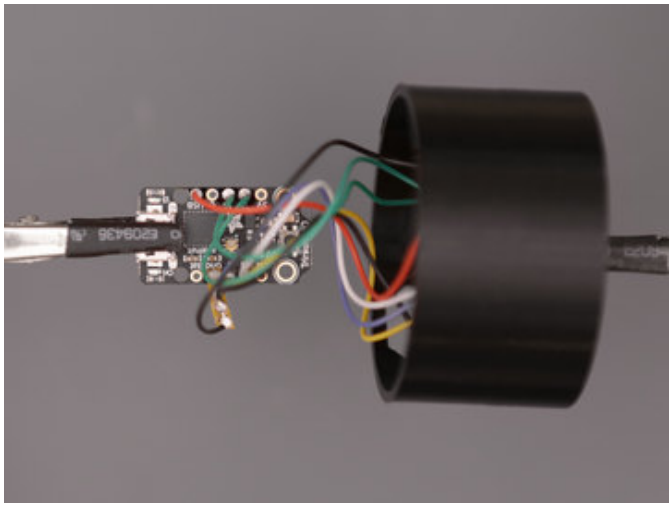
If you haven't already, gently ned the legs back so they don't make contact with the Trinket once mounted in place.



Attach washer

Now we can place the washer and screw back on to the Rotary Encoder. Hold it in place while screwing the hex nut back onto the steam.



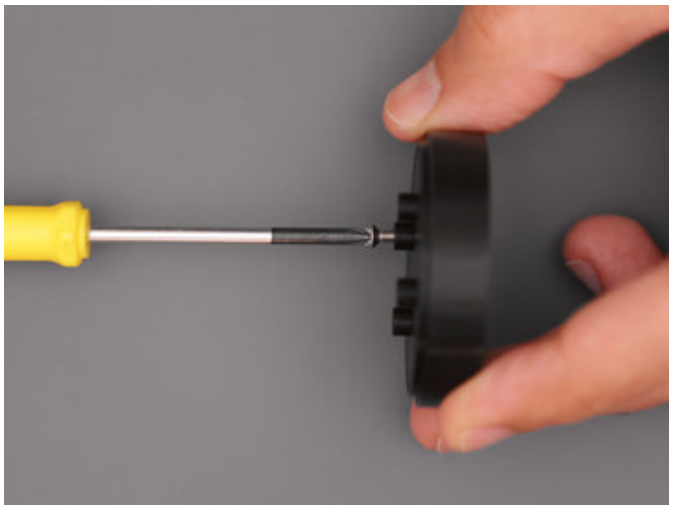


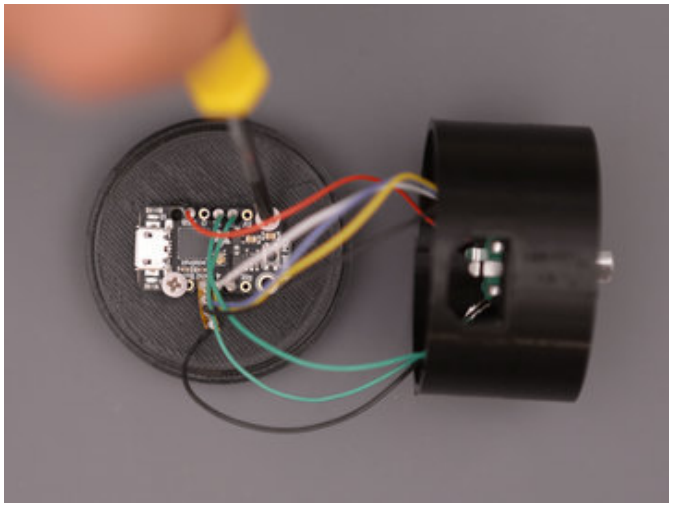
Solder Rotary to Trinket

With the Rotary mounted we can finally move on to connect it to the Trinket.

Align the Rotary assembly next to the Trinket with a pair of third helping hands.

Reference the circuit diagram and solder each connection for the NeoPixel ring and Rotary to the Trinket M0.





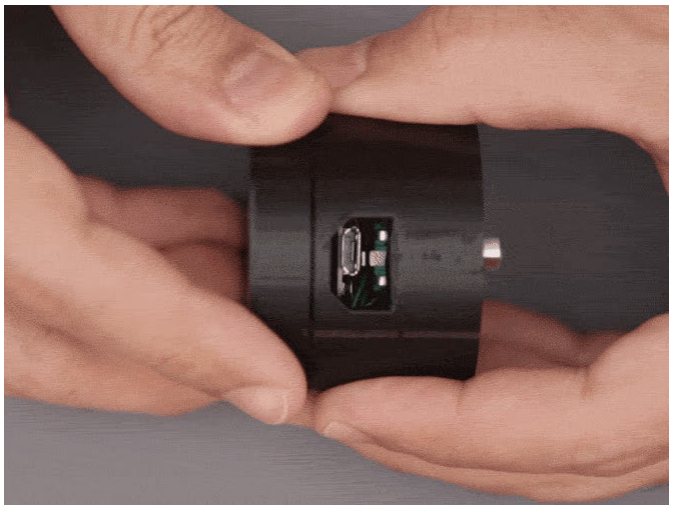
Mount Trinket

Now we can mount the Trinket to the **dial-lid** part. We'll use **M2.5x5mm** long screws to securely mount the Trinket.

First, we'll create threads for the mounts by fastening the screws to before mounting the Trinket. Make sure the screws fasten straight to properly align the Trinket mounting holes.

Wire Manegement

Some heat shrink tubing or kapton tape can be used to keep the wires nice and tidy.



Align USB port

Now we can attach the Trinket to to rest of the enclosure. Snap fit the **dial-lid** onto the **dial-mid** by attaching it an angle.

Insert the back side of the lid (opposite side of the USB opening) first and then rotate the lid until the USB port on the Trinket and the opening on the enclosure align.

Attach Dial

Align the printed dial to the notch on the Rotary Encoders steam and press fit until it pushes all the way into the part.

Make sure to apply even pressure to the dial to avoid any wobble on the cover while turning the dial.

Complete!

Now we can fit our USB cable into the Trinket and setup any key strokes to assign to the media dial!

We can additional attach rubber feet or even add tac to prevent the dial from moving while in use.

If the LED ring lights up in the opposite direction while turning, you'll mostly likely need to swap the two wires on the three leg side of the Rotary Encoder.