Ocean Epoxy Resin Lightbox with RGB LED Matrix Image Scroller

Created by Erin St Blaine
## Guide Contents

**Overview**
- Difficulty Level  
- Parts  
- Additional Materials Used

**Circuit Diagram**

**CircuitPython on Feather M4 Express**
- Set up CircuitPython Quick Start!
- Further Information

**CircuitPython Code**
- Installing or upgrading CircuitPython  
- Download the Adafruit CircuitPython Library Bundle  
- Required Libraries  
- Upload Code  
- Double Check

**Code Explanation**
- Understanding the code

**Add Custom Images**

**Resin Ocean**
- Supplies  
- How Much Resin?

**Electronics Assembly**
- Assemble the FeatherWing  
- Stack the Feather M4  
- Final Assembly  
- Troubleshooting

**Final Assembly**
Overview

See the line where the sky meets the sea
It calls me
And no one knows, how far it goes
If the wind in my sail on the sea stays behind me
One day I'll know
If I go there's just no telling how far I'll go

- Moana: How Far I'll Go - (Lin Manuel Miranda)

Explore the depths of your own personal ocean. Epoxy resin, alcohol inks, sand, and seashells come together to make a magical luminous ocean experience for your wall or desk. Each piece you create will come out unique, colorful, and beautiful.

Difficulty Level

The electronics assembly is pretty simple - the only soldering you'll need to do is stacking and assembling the Feather and FeatherWing. The LED matrix plugs into the FeatherWing and into the wall using connectors, so doesn't need to be soldered at all.

Epoxy resin is tricky by nature. Temperature, humidity, and dust need to be managed with any resin project. However, as resin projects go, I've found this to be about the easiest one I've ever done -- and also the hardest to mess up. Stray bubbles or dust motes in your resin actually add depth and interest to this particular project - more bubbles make the ocean waves look more realistic! And as you'll see, mixing and pouring all the pretty colors is incredibly satisfying.
The software and coding requires a little bit of setup if you haven't used CircuitPython before. Once you're set up, you can copy and paste our code and images or dig in and customize with your own artwork.

Parts

Your browser does not support the video tag.

64x32 RGB LED Matrix - 4mm pitch
Bring a little bit of Times Square into your home with this sweet 64 x 32 square RGB LED matrix panel. These panels are normally used to make video walls, here in New York we see them...
$39.95
In Stock
Add to Cart

Adafruit Feather M4 Express - Featuring ATSAMD51
It's what you've been waiting for, the Feather M4 Express featuring ATSAMD51. This Feather is fast like a swift, smart like an owl, strong like a ox-bird (it's half ox,...
$22.95
In Stock
Add to Cart

Adafruit RGB Matrix Featherwing Kit
Ahoy! It's time to create a dazzling light up project with our new RGB Matrix FeatherWing. Now you can quickly and easily create...
$7.50
In Stock
Add to Cart

5V 4A (4000mA) switching power supply - UL Listed
Need a lot of 5V power? This switching supply gives a clean regulated 5V output at up to 4 Amps (4000mA). 110 or 240 input, so it works in any country. The plugs are "US...
$14.95
In Stock
Add to Cart

USB cable - USB A to Micro-B
This here is your standard A to micro-B USB cable, for USB 1.1 or 2.0. Perfect for connecting a PC to your Metro, Feather, Raspberry Pi or other dev-board or...
$2.95
In Stock
Add to Cart
If you want an easy on/off switch for your project, I recommend this one - it plugs right inline into your power cord.

1 x **Power Switch**

In-line power switch for 2.1mm barrel jack

---

I also used rare earth magnets to fasten the resin panel to the shadow box.

4 x **Magnets**

High-strength 'rare earth' magnet

---

**Additional Materials Used**

My ocean is 12"x12". If you're making a similar sized ocean you'll need about 2 cups of table top resin or casting resin, alcohol inks and mica powder for color, and a piece of clear acrylic or glass to act as your "canvas". You'll also need gloves, mixing cups and stirring sticks and various other accessories. I've made an Amazon shopping cart ([https://adafruit.it/MDO](https://adafruit.it/MDO)) with everything I use when I'm pouring resin.

You'll also need an enclosure for the electronics. I built my own out of spare plastic sheeting I had in my shop, but you could also use a clear plastic tray or a shadow box ([https://adafruit.it/MEZ](https://adafruit.it/MEZ)) frame for your project. Just be sure it's at least 2-3 inches deep to give enough space for the electronics and a little extra room for diffusion.
The FeatherWing comes with all the connectors you need. We'll solder the headers onto the Feather M4 and FeatherWing, then plug the RGB Matrix in with the included connectors.
CircuitPython on Feather M4 Express

CircuitPython ([https://adafru.it/tB7](https://adafru.it/tB7)) is a derivative of MicroPython ([https://adafru.it/BeZ](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 (which appears as a USB flash drive when you plug the board into your computer via a USB cable) to iterate.

The following instructions will show you how to install CircuitPython. If you've already installed CircuitPython but are looking to update it or reinstall it, the same steps work for that as well!

Set up CircuitPython Quick Start!

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

https://adafru.it/Emh

Click the link above and download the latest UF2 file.

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

Plug your Feather M4 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.

Double-click the Reset button next to the USB connector on your board, and you will see the NeoPixel RGB LED turn green. If it turns red, check the USB cable, try another USB port, etc. **Note:** The little red LED next to the USB connector will pulse red. That's ok!

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 FEATHERBOOT.

Drag the adafruit_circuitpython_etc.uf2 file to FEATHERBOOT.

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

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

Further Information

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

Installing or upgrading CircuitPython

You should ensure you have CircuitPython 5.3 or greater on your board. Plug your board in with a known good data + power cable (not the cheesy USB cable that comes with USB power packs, they are power only). You should see a new flash drive pop up.

If the drive is CIRCUITPY, then open the boot_out.txt file to ensure the version number is 5.3 or greater.

Adafruit CircuitPython 5.3.0 on 2020-04-29; Adafruit Feather M4 Express with samd51j19

If the version is less than 5 -or- you only get a drive named FEATHERBOOT then follow the Feather M4 guide (https://adafru.it/CJN) on installing CircuitPython (https://adafru.it/CVs).

Download the Adafruit CircuitPython Library Bundle

In order to run the code, we’ll need to download a few libraries. Libraries contain code to help interface with hardware a lot easier for us.

Use the Feather M4 page on Installing Libraries (https://adafru.it/HDo) to get the library that matches the major version of CircuitPython you are using noted above.

The green button below links to a file containing all the libraries available for CircuitPython. To run the code for this project, we need a few of the libraries in the list below. Unzip the library bundle and search for the libraries. Drag and drop the files into a folder named lib on the CIRCUITPY drive (create the folder if it is not already on the Feather M4).

https://adafru.it/ENC

https://adafru.it/ENC

Required Libraries

- adafruit_bus_device (directory)
- adafruit_display_shapes (directory)
- adafruit_imageload (directory)

You’ll also need the .bmp files. You can add your own files or customize ours. If you’re adding your own .bmp files, be sure they’re 64 pixels wide. I had the most success when I saved as 8 bit color-indexed files.

https://adafru.it/ME1
Once we have all the files we need, a directory listing will look similar to below as far as files and directories.

![Directory Listing](https://adafru.it/MEl)

**Upload Code**

Click on the Download: Project Zip link below to grab the project files in a zip file directly from GitHub. Place the `code.py` file and bitmap graphics files onto the `CIRCUITPY` main (root) directory. The code will run properly when all of the files have been uploaded including libraries.

Use any text editor or favorite IDE to modify the code. We suggest using Mu as noted above.

```python
import math
import time
import random
import adafruit_imageload.bmp
import board
import displayio
import framebufferio
import rgbmatrix
import ulab
```

"""
RGB Matrix Ocean Scroller
Adafruit invests time and resources providing this open source code.
Please support Adafruit and open source hardware by purchasing products from Adafruit!
Written by Jeff Epler & Limor Fried for Adafruit Industries
Copyright (c) 2019-2020 Adafruit Industries
Licensed under the MIT license.
All text above must be included in any redistribution.
"""
displayio.release_displays()

class Reshader:
    '''reshader fades the image to mimic brightness control'''
    def __init__(self, palette):
        self.palette = palette
        ulab_palette = ulab.numpy.zeros((len(palette), 3))
        for i in range(len(palette)):
            rgb = int(palette[i])
            ulab_palette[i, 2] = rgb & 0xff
            ulab_palette[i, 1] = (rgb >> 8) & 0xff
            ulab_palette[i, 0] = rgb >> 16
        self.ulab_palette = ulab_palette

    def reshade(self, brightness):
        '''reshader'''
        palette = self.palette
        shaded = ulab.numpy.array(self.ulab_palette * brightness, dtype=ulab.numpy.uint8)
        for i in range(len(palette)):
            palette[i] = tuple(shaded[i])

    def do_crawl_down(image_file, *,
        speed=12, weave=4, pulse=.5,
        weave_speed=1/6, pulse_speed=1/7):
        # pylint:disable=too-many-locals
        '''function to scroll the image'''
        the_bitmap, the_palette = adafruit_imageload.load(image_file,
            bitmap=displayio.Bitmap,
            palette=displayio.Palette)
        shader = Reshader(the_palette)

        group = displayio.Group()
        tile_grid = displayio.TileGrid(bitmap=the_bitmap, pixel_shader=the_palette)
        group.append(tile_grid)
        display.show(group)

        start_time = time.monotonic_ns()
        start_y = display.height   # High enough to be "off the top"
        end_y = -the_bitmap.height     # Low enough to be "off the bottom"

        # Mix up how the bobs and brightness change on each run
        r1 = random.random() * math.pi
        r2 = random.random() * math.pi

        y = start_y
        while y > end_y:
            now = time.monotonic_ns()
            y = start_y - speed * ((now - start_time) / 1e9)
            group.y = round(y)

            # wave from side to side
            group.x = round(weave * math.cos(y * weave_speed + r1))

            # Change the brightness
            if pulse > 0.
if pulse > 0:
    shader.reshade((1 - pulse) + pulse * math.sin(y * pulse_speed + r2))

display.refresh(minimum_frames_per_second=0, target_frames_per_second=60)

def do_pulse(image_file, *, duration=4, pulse_speed=1/8, pulse=.5):
    '''pulse animation'''
    the_bitmap, the_palette = adafruit_imageload.load(image_file,
        bitmap=displayio.Bitmap,
        palette=displayio.Palette)

    shader = Reshader(the_palette)

    group = displayio.Group()
    tile_grid = displayio.TileGrid(bitmap=the_bitmap, pixel_shader=the_palette)
    group.append(tile_grid)
    group.x = (display.width - the_bitmap.width) // 2
    group.y = (display.height - the_bitmap.height) // 2
    display.show(group)

    start_time = time.monotonic_ns()
    end_time = start_time + int(duration * 1e9)

    now_ns = time.monotonic_ns()
    while now_ns < end_time:
        now_ns = time.monotonic_ns()
        current_time = (now_ns - start_time) / 1e9

        shader.reshade((1 - pulse) - pulse
                        * math.cos(2*math.pi*current_time*pulse_speed)**2)

        display.refresh(minimum_frames_per_second=0, target_frames_per_second=60)

matrix = rgbmatrix.RGBMatrix(
    width=64, height=32, bit_depth=5,
    addr_pins=[board.A5, board.A4, board.A3, board.A2],
    clock_pin=board.D13, latch_pin=board.D0, output_enable_pin=board.D1)

display = framebufferio.FramebufferDisplay(matrix, auto_refresh=False)

# Image playlist - set to run randomly. Set pulse from 0 to .5
while True:
    show_next = random.randint(1, 5) # change to reflect how many images you add
    if show_next == 1:
        do_crawl_down("/ray.bmp")
    elif show_next == 2:
        do_crawl_down("/waves1.bmp", speed=7, weave=0, pulse=.35)
    elif show_next == 3:
        do_crawl_down("/waves2.bmp", speed=9, weave=0, pulse=.35)
    elif show_next == 4:
        do_pulse("/heart.bmp", duration=4, pulse=.45)
    elif show_next == 5:
        do_crawl_down("/dark.bmp")
Double Check

See the directory listing above and double check that you have all the files listed to make this project function. If any are missing or in an incorrect directory, move them so they’re in the right places.
Understanding the code

There is a lot going on to bring these images to life on the LED matrix, so let's break things down a little bit.

Begin by importing the necessary modules and ensuring that we can create our display:

```python
import math
import time
import random
import adafruit_imageload.bmp
import board
import displayio
import framebufferio
import rgbmatrix
import ulab

displayio.release_displays()
```

RGBMatrix doesn’t have a brightness control, so the Reshader class uses ulab (https://adafruit.it/KaJ) to quickly apply a brightness value from 0.0 to 1.0 to an original palette, updating the palette accordingly. This code works pretty quickly, especially when you make sure your images have a palette of just a few dozen colors.

```python
class Reshader:
    '''reshader fades the image to mimic brightness control'''
    def __init__(self, palette):
        self.palette = palette
        ulab_palette = ulab.numpy.zeros((len(palette), 3))
        for i in range(len(palette)):
            rgb = int(palette[i])
            ulab_palette[i, 2] = rgb & 0xff
            ulab_palette[i, 1] = (rgb >> 8) & 0xff
            ulab_palette[i, 0] = rgb >> 16
        self.ulab_palette = ulab_palette

    def reshape(self, brightness):
        ''' reshader '''
        palette = self.palette
        shaded = ulab.numpy.array(self.ulab_palette * brightness, dtype=ulab.numpy.uint8)
        for i in range(len(palette)):
            palette[i] = tuple(shaded[i])
```

do_crawl_down makes an image move from the top of the screen to the bottom. Here are what all the
parameters mean: **image_file** is the filename of an image. Images up to 64x64 pixels work best. **speed** is the down speed in pixels per second. **weave** gives the amount the image weaves left and right, 0 for no weave. **weave_speed** controls how quickly the weave goes. **pulse** controls how much the brightness changes, values from 0 (no brightness change) to 0.5 work well.

By working with time in integer nanoseconds, we avoid rounding errors that occur when working with time in floating-point seconds. It makes the code a bit more complicated, but avoids the animation becoming "jerky" after running continuously for more than a day or so.

```python
def do_crawl_down(image_file, *,
    speed=12, weave=4, pulse=.5,
    weave_speed=1/6, pulse_speed=1/7):
    '''function to scroll the image'''
    the_bitmap, the_palette = adafruit_imageload.load(
        image_file,
        bitmap=displayio.Bitmap,
        palette=displayio.Palette)

    shader = Reshader(the_palette)

    group = displayio.Group()
    tile_grid = displayio.TileGrid(bitmap=the_bitmap, pixel_shader=the_palette)
    group.append(tile_grid)
    display.show(group)

    start_time = time.monotonic_ns()
    start_y = display.height   # High enough to be "off the top"
    end_y = -the_bitmap.height     # Low enough to be "off the bottom"

    # Mix up how the bobs and brightness change on each run
    r1 = random.random() * math.pi
    r2 = random.random() * math.pi

    y = start_y
    while y > end_y:
        now = time.monotonic_ns()
        y = start_y - speed * ((now - start_time) / 1e9)
        group.y = round(y)

        # wave from side to side
        group.x = round(weave * math.cos(y * weave_speed + r1))

        # Change the brightness
        if pulse > 0:
            shader.reshade((1 - pulse) + pulse * math.sin(y * pulse_speed + r2))

        display.refresh(minimum_frames_per_second=0, target_frames_per_second=60)
```

**do_pulse** omits the movement part of do_crawl_down, so it's basically a simplified version of do_crawl_down.
def do_pulse(image_file, *, duration=4, pulse_speed=1/8, pulse=.5):
    '''pulse animation'''
    the_bitmap, the_palette = adafruit_imageload.load(
        image_file,
        bitmap=displayio.Bitmap,
        palette=displayio.Palette)

    shader = Reshader(the_palette)

    group = displayio.Group()
    tile_grid = displayio.TileGrid(bitmap=the_bitmap, pixel_shader=the_palette)
    group.append(tile_grid)
    group.x = (display.width - the_bitmap.width) // 2
    group.y = (display.height - the_bitmap.height) // 2
    display.show(group)

    start_time = time.monotonic_ns()
    end_time = start_time + int(duration * 1e9)

    now_ns = time.monotonic_ns()
    while now_ns < end_time:
        now_ns = time.monotonic_ns()
        current_time = (now_ns - start_time) / 1e9

        shader.reshade((1 - pulse) - pulse
                        * math.cos(2*math.pi*current_time*pulse_speed)**2)

        display.refresh(minimum_frames_per_second=0, target_frames_per_second=60)

Create the display object. Check out our dedicated guide for rgbmatrix (https://adafruit.it/L7b) to find pinouts for other boards besides the Feather M4 Express!

matrix = rgbmatrix.RGBMatrix(
    width=64, height=32, bit_depth=5,
    addr_pins=[board.A5, board.A4, board.A3, board.A2],
    clock_pin=board.D13, latch_pin=board.D0, output_enable_pin=board.D1)
display = framebufferio.FramebufferDisplay(matrix, auto_refresh=False)

Choose randomly from 5 different effects. By using different images and different combinations of parameters, there’s more variety. Why not try picking your own combinations, and see what you like best! Head to the next page for more information about customizing this with your own images.
while True:
    show_next = random.randint(1, 5) #change to reflect how many images you add
    if show_next == 1:
        do_crawl_down("/ray.bmp")
    elif show_next == 2:
        do_crawl_down("/waves1.bmp", speed=7, weave=0, pulse=.35)
    elif show_next == 3:
        do_crawl_down("/waves2.bmp", speed=9, weave=0, pulse=.35)
    elif show_next == 4:
        do_pulse("/heart.bmp", duration=4, pulse=.45)
    elif show_next == 5:
        do_crawl_down("/dark.bmp")

Add Custom Images

This code scrolls .bmp images from the bottom of the screen to the top. I've placed the screen upside-down inside my project so it looks like the images are scrolling downwards.

The images also fade in and out, and can weave back and forth for a more organic feel. The code has variables to make it easy to customize the way each of your images behaves.

Look at the bottom of the code to find the `while True` section.

```python
while True:
    show_next = random.randint(1, 5) #change to reflect how many images you add
    if show_next == 1:
        crawl_down(display, "/ray.bmp")
    elif show_next == 2:
        crawl_down(display, "/waves1.bmp", speed=7, weave=0, pulse=.35)
    elif show_next == 3:
        crawl_down(display, "/waves2.bmp", speed=9, weave=1, pulse=.35)
    elif show_next == 4:
        pulse(display, "/heart.bmp", duration=4, pulse=.45)
    elif show_next == 5:
        crawl_down(display, "/dark.bmp")
```

To customize your image scroller:

1. Save your 64px wide .bmp images at the root of the CIRCUITPY drive along with the sample images we've included.
2. Update this part of your code so the .bmp names match the names of your files.
3. If you have more or fewer than 5 files, change the `random.randint(1, 5)` line to reflect the number of images (or separate instances of the same image) you want to add. For example, if you have only 3 files, it should read `random.randint(1, 3)`.
4. Customize the behavior of each image by modifying the `speed`, `weave`, `pulse`, and `duration` variables. The `pulse` variable should be between 0 and 0.5. A value of .45 seems to work well for an image that doesn't fade out completely.
Gather all your supplies. Once the resin is mixed you'll have a limited amount of time to pour before it starts setting up, so it's good to have everything close to hand.

**Supplies**

You can get a lot of this stuff at your local hardware or craft store, or here is my [Resin Art Shopping Cart](https://adafru.it/MDO) at Amazon.com if you want to order everything at once.

- 2-part Epoxy Resin - Table Top resin or casting resin will work for smaller projects
- Alcohol inks for coloring in at least 3-4 colors - I used blue, sapphire, cerulean, teal, and green
- Mica powder in white for the surf
- Protective gloves
- A graduated measuring cup
- A larger plastic cup for mixing
- Stirring sticks
- 5-6 smaller cups for mixing individual colors (I used dixie cups, similar small cups are great)
- Glass or plastic or other translucent "canvas" - mine is a 12x12 piece of 1/4" acrylic
- Mirror effect spray paint
- Sand paper
- Heat gun
- Paper towels for cleanup
- Plastic drop cloths
- Something to place under your canvas to elevate it off the table
- White sand
- Seashells or other 3d decor

You'll want to work in a dust-free environment, with low humidity and a temperature range that matches the recommended working temperature on your resin. This brand recommends 75-85F, so I also have a space heater and an air purifier in my work area.

If your acrylic is perfectly clear, it'll be easy to see through the ocean waves and spot the electronics inside. To prevent this, I prepped my acrylic by sanding it with a piece of sandpaper on both sides until it
was nice and cloudy.

Then I sprayed one side with 1-2 coats of Mirror Effect spray paint. This will create a semi-translucent one-way mirror effect. It's one of my favorite ways to create a soft diffusion with LED lights. Go easy on this - you can always add another layer of paint later on to the back of your piece if it's too transparent.

Place the acrylic on your plastic-covered table, with a piece of wood or something underneath so the drips can fall off the edges easily onto your plastic.

How Much Resin?

The amount to use will depend on the size of your piece. My resin bottle said that 1 gallon would cover 12 square feet at 1/8" thick, so for 1 square foot I used 1/12 of a gallon, or 16 ounces. That's two cups, which makes measuring super easy - I used one cup of part A and one cup of part B.

Follow the directions on your resin bottle, paying attention to working temperatures and mixing times. I'm using table top resin, which wants to be mixed for five full minutes. Don't under-mix, or your resin won't cure evenly.

Mix it by hand with a stirring stick, and mix slowly to minimize the amount of bubbles that get pulled into your resin. Scrape the sides and bottom of the cup and your stirring stick while you mix.

Set out your small mixing cups. Have one for each alcohol-ink color you're using, one for the white mica powder, and one for the sand.

Pour about 1 inch of resin into each of your small cups. Make sure to keep about 1/4 of the resin in reserve in the main stirring cup - we'll need some clear resin too so don't color it all.

Mix in about 1/4 teaspoon of white mica powder into the cup. Mix it well. Add more if it doesn't look opaque enough.

Add a few drops of alcohol inks to each of your color cups. Don't go overboard - we want translucent color - but make them nice and bright.

Stir about 1/4 c of white sand into your remaining cup. Add enough resin to the sand that it's sticky and gloppy without being runny. We want the sand to pile up a bit.
Start in a corner of your piece with the sand. Pile it up until it looks beachy.

Pour the colored resin in waves, leaving the area by the beach empty for now - we'll fill that part with clear resin in a bit.

Use your gloved fingertips to gently mix the colors until you have a nice ocean-y gradient.

Pour some of your clear resin into the gap and mix that in as well.

Wait about 2 minutes to let the resin just start to set up. If bubbles form on the surface, use your heat gun about 6-8 inches above the surface to gently heat the air and pop the bubbles.

We want the resin to be a bit firm so the white "surf" floats on the top instead of mixing into the blues and greens.

While you're waiting, dip your sea shells into the clear resin to coat them. This will make them stick to your piece better, and the extra resin will pool around their base, making them look like they're sitting in wet sand. Place them firmly into the sand-resin mixture.

After your 2 minutes are up, pour a couple stripes of white resin where you want the waves to go. Be sparing with the white! You can always add more, but it's easy to get too much.

Now for the fun part: grab your hair drier and turn it to "high". Blow warm air across the resin and watch the ocean waves appear.

Let it pour off the edges a bit, and make sure your whole piece is covered.
I let my piece dry overnight, then mixed up a little more resin and added a second wave on top of the first wave.

Let your piece cure for the recommended time specified in the directions - usually 24-48 hours. Trim off any drips on the back of your piece with a pair of flush-cut snips.
The Feather M4 Express and RGB FeatherWing make this project really easy. We'll assemble the FeatherWing, stack it on top of the Feather, and then plug in the rest of the connections.

Assemble the FeatherWing

Find all the pieces for your FeatherWing - they are all included with the board. First we'll solder on the headers, then the rest of the connectors.

If you haven't soldered headers before, it seems scary but is really pretty straightforward. Check our our Guide to Excellent Soldering (https://adafru.it/dxy) for tips.

We'll be inserting the headers into the outer rows of holes on either side of the FeatherWing, so we need to make sure we have the right number of pins. One of them is right. The other needs to be shortened - I had to break 4 pins off my second header to get it into shape.
The easiest way to neatly line up the headers is to use a solderless breadboard (https://adafru.it/MDP) to hold them. Put the long end of the pins into the breadboard, and place the FeatherWing face up on the short end of the pins as shown. Solder all the pins in place.
Next we'll add the screw terminal. This will connect power from the RGB Matrix to the board.

Look closely to be sure you've got it aligned correctly. The slope faces toward the outside of the board.

Flip it over and solder the underside of the pins to the copper pads.
Next, solder the DC power jack onto the other end of the FeatherWing.
Finally, add the ribbon cable connector. Look for the cutout on the side of the connector and align it with the cutout symbol etched on the board to be sure you've got it the right way around.

Finally, grab your Feather M4 Express and place it onto the long side of the header pins you soldered in step 1. Solder the pins in place on the Feather M4.

That's it for the soldering!
Find the power connector that came with your RGB panel. Cut off the screw terminal connectors with wire cutters and strip just a bit of shielding from each wire.

Insert the wires into your screw terminal on your FeatherWing, being sure to line up the red wire with + and the black wire with -, as shown. Screw the ports tightly down on the wires with a small screwdriver.

Plug the power cable from the screw terminal into the plug on the back of your matrix, and plug the ribbon cable in on both ends. Plug in your power supply. Flip your panel over and be sure it's working.

Troubleshooting

If your panel doesn't light up, here are some things to try:

- Re-upload your software, or load some test code. Really with this build, the software is the most likely
culprit for mishaps.

- Are there any "bridged" solder joints on your board? Be sure none of the pins are touching each other.
- Try plugging the ribbon cable into the other port on the back of the matrix.
- Wiggle all your connections, especially the screw terminal connections which can sometimes be tricky to get tight.

If none of this helps, head over to the [RGB FeatherWing guide](https://adafru.it/MDQ) for more suggestions.
Final Assembly

Drill a hole in the side or the back of your frame to let the power cord through. I used a rotary tool for this.

Use hot glue to secure your power cord, Feather, and LED matrix in place inside your frame. I found it worked well to glue along the top edge of the matrix and fix that to the top of the frame, since it doesn't sit flat inside the box.
To attach the resin piece to the frame, I used four rare earth magnets and four spare nuts from a jar in my shop. I used E6800 glue to secure the bolts to the inside of the frame, about 1/4" down from the top.
Measure between the bolts and place the magnets on the back of the acrylic where they'll meet up and fit snugly. Use a generous amount of glue and let it dry completely - these magnets are STRONG and will overrule weaker glue.

If you are adding the inline on/off switch, plug it into the Feather and the power cord and mount it someplace convenient on the edge of your box - you may need to drill another hole depending on your setup.

Hang it on a wall with picture hanging hardware or set it on your desk like a picture frame. Enjoy your beautiful, magical, bioluminescent ocean.