LED Protest Sign

Created by John Park

https://learn.adafruit.com/led-protest-sign

Last updated on 2021-12-08 04:57:48 PM EST
# Table of Contents

**Overview**
- Parts 5
- Additional Materials 8
- 3D Printing 8
- Tools 9

**Using Feather M4 Express** 9

**CircuitPython on Feather M4 Express** 10
- Set up CircuitPython Quick Start! 10
- Further Information 12

**Using MatrixPortal** 12

**Prep the MatrixPortal** 14
- Power Prep 14
- Power Terminals 15
- Panel Power 15
- Board Connection 16

**Install CircuitPython** 17
- Set up CircuitPython Quick Start! 17
- Further Information 17

**CircuitPython Libraries** 19
- The Adafruit CircuitPython Library Bundle 21
- Downloading the Adafruit CircuitPython Library Bundle 21
- The CircuitPython Community Library Bundle 22
- Downloading the CircuitPython Community Library Bundle 22
- Understanding the Bundle 22
- Example Files 23
- Copying Libraries to Your Board 24
- Understanding Which Libraries to Install 24
- Example: ImportError Due to Missing Library 27
- Library Install on Non-Express Boards 28
- Updating CircuitPython Libraries and Examples 28

**Code the Protest Sign** 29
- Libraries 29
- Text Editor 29
- Code.py 29
- Image Files 30

**Build the Sign** 31
- Power 32
- Battery Pack 33
- Add Handle 34
- Feather Prep 35
- FeatherWing Prep 37
- Connect to Sign 38
- Add Batteries 39
Overview

Build a protest/demonstration sign that can be seen at night. This project makes it simple to use your own graphics on the sign -- it's as easy as dragging and dropping .bmp image files into a folder.

CircuitPython running on the Feather M4 with the RGB Matrix FeatherWing drives any of our RGB LED matrix panels.

You can use the large 6mm pitch version and 3D print the provided brackets to hold LED diffusing plastic. Screw a yardstick or other thin strip of wood to the panel and you can go out and march.

Parts

Adafruit Matrix Portal - CircuitPython Powered Internet Display

Folks love our wide selection of RGB matrices and accessories, for making custom colorful LED displays... and our RGB Matrix Shields...

https://www.adafruit.com/product/4745
64x32 RGB LED Matrix - 6mm pitch
Bring a little bit of Times Square into your home with this sweet 64x32 square RGB LED matrix panel. These panels are normally used to make video walls, here in New York we see them on...
https://www.adafruit.com/product/2276

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,...
https://www.adafruit.com/product/3857

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...
https://www.adafruit.com/product/3036

8 x AA battery holder with 5.5mm/2.1mm Plug and On/Off Switch
Make a portable power brick with plenty of juice! Use Alkaline AA's for a 12V 3000-4000mAh power supply, or rechargeable NiMH for 2000mAh 9.6V supply. Either one is good for running...
https://www.adafruit.com/product/875
UBEC DC/DC Step-Down (Buck) Converter - 5V @ 3A output
Your power supply problems just got SOLVED! This little circuit board may look tiny but inside is a high efficiency DC/DC step-down converter which can output up to 3 Amp at 5V without...
https://www.adafruit.com/product/1385

Panel Mount 2.1mm DC barrel jack
This power jack is designed to easily attach to a panel up to 8mm thick (0.315" or 5/16") and fit 2.1mm power plugs snugly and securely. Perfect for adding a power connector...
https://www.adafruit.com/product/610

5.5 / 2.1mm Barrel Connector - DC Power Plug
DIY? How about D-I-Wire! This barrel jack plug is great for adding a common power connector to the end of your wires. The jack is compatible with 5.5mm barrel jacks that have a...
https://www.adafruit.com/product/3310

Heat Shrink Pack
Heat shrink is the duct tape of electronics, it keeps your stuff all safe and kept together. Especially when wiring and soldering, use heat shrink to add mechanical strength to cables....
https://www.adafruit.com/product/344
Black Nylon Machine Screw and Stand-off Set – M2.5 Thread
Totaling 380 pieces, this M2.5 Screw Set is a must-have for your workstation. You'll have enough screws, nuts, and hex standoffs to fuel your maker...
https://www.adafruit.com/product/3299

Additional Materials

In addition to the parts above, you'll need:

- 8 ea. AA NiMH rechargeable batteries
- A length of 2" x 1/4" wood for a handle, such as a yard stick
- 6 ea. M3 x 8mm screws (https://adafruit.it/LHb) for securing the brackets (these can be the typical Phillips head, but I really like hex drive screws)
- 2 ea. M3 x 12mm screws (https://adafruit.it/LHc) for attaching the handle
- Sheet of diffusing acrylic 385mm x 190mm x 3mm such as Chemcast Black LED Acrylic (https://adafruit.it/IVf)
- Velcro or double stick foam tape to secure the battery holder
- Zip ties

3D Printing
For the brackets and Feather cover, you'll need access to a 3D printer and some PLA or similar filament.
Tools

You'll need:

- screwdriver
- soldering iron and solder
- wire strippers
- hand drill

Using Feather M4 Express

You can build this project using the Feather M4 Express. For a detailed list of all parts needed for this project, check out the Overview.

64x32 RGB LED Matrix - 6mm pitch
Bring a little bit of Times Square into your home with this sweet 64x32 square RGB LED matrix panel. These panels are normally used to make video walls, here in New York we see them on...
https://www.adafruit.com/product/2276

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...
https://www.adafruit.com/product/3036
CircuitPython on Feather M4 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.

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 :) 

Download the latest version of CircuitPython for this board via CircuitPython.org 
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://adafru.it/Amd).

Using MatrixPortal

You can build this project with an all-in-one Matrix Portal board, its definitely the easiest and least-expensive way to go about it.

You will need a matrix portal, matrix, and USB C power/data cable

Adafruit Matrix Portal - CircuitPython Powered Internet Display
Folks love our wide selection of RGB matrices and accessories, for making custom colorful LED displays... and our RGB Matrix Shields...
https://www.adafruit.com/product/4745
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...
https://www.adafruit.com/product/2278

USB Type A to Type C Cable - approx 1 meter / 3 ft long
As technology changes and adapts, so does Adafruit. This USB Type A to Type C cable will help you with the transition to USB C, even if you're still...
https://www.adafruit.com/product/4474
Prep the MatrixPortal

Power Prep

The MatrixPortal supplies power to the matrix display panel via two standoffs. These come with protective tape applied (part of our manufacturing process) which MUST BE REMOVED!

Use some tweezers or a fingernail to remove the two amber circles.
Power Terminals
Next, screw in the spade connectors to the corresponding standoff.

- red wire goes to +5V
- black wire goes to GND

Panel Power
Plug either one of the four-conductor power plugs into the power connector pins on the panel. The plug can only go in one way, and that way is marked on the board's silkscreen.
Board Connection

Now, plug the board into the left side shrouded 8x2 connector as shown. The orientation matters, so take a moment to confirm that the white indicator arrow on the matrix panel is oriented pointing up and right as seen here and the MatrixPortal overhangs the edge of the panel when connected. This allows you to use the edge buttons from the front side.

Check nothing is impeding the board from plugging in firmly. If there's a plastic nub on the matrix that's keeping the Portal from sitting flat, cut it off with diagonal cutters.
Install CircuitPython

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/Nte

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 MatrixPortal 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 (indicated by the green arrow) on your board, and you will see the NeoPixel RGB LED (indicated by the magenta arrow) turn green. If it 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 MATRIXBOOT.

Drag the adafruit_circuitpython_etc.uf2 file to MATRIXBOOT.

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

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

CircuitPython Libraries

As CircuitPython development continues and there are new releases, Adafruit will stop supporting older releases. Visit https://circuitpython.org/downloads to download the latest version of CircuitPython for your board. You must download the CircuitPython Library Bundle that matches your version of CircuitPython. Please update CircuitPython and then visit https://circuitpython.org/libraries to download the latest Library Bundle.
Each CircuitPython program you run needs to have a lot of information to work. The reason CircuitPython is so simple to use is that most of that information is stored in other files and works in the background. These files are called libraries. Some of them are built into CircuitPython. Others are stored on your CIRCUITPY drive in a folder called lib. Part of what makes CircuitPython so great is its ability to store code separately from the firmware itself. Storing code separately from the firmware makes it easier to update both the code you write and the libraries you depend.

Your board may ship with a lib folder already, it's in the base directory of the drive. If not, simply create the folder yourself. When you first install CircuitPython, an empty lib directory will be created for you.

CircuitPython libraries work in the same way as regular Python modules so the Python docs (https://adafru.it/rar) are an excellent reference for how it all should work. In Python terms, you can place our library files in the lib directory because it's part of the Python path by default.

One downside of this approach of separate libraries is that they are not built in. To use them, one needs to copy them to the CIRCUITPY drive before they can be used. Fortunately, there is a library bundle.

The bundle and the library releases on GitHub also feature optimized versions of the libraries with the .mpy file extension. These files take less space on the drive and have a smaller memory footprint as they are loaded.

Due to the regular updates and space constraints, Adafruit does not ship boards with the entire bundle. Therefore, you will need to load the libraries you need when you begin working with your board. You can find example code in the guides for your board that depends on external libraries.

Either way, as you start to explore CircuitPython, you'll want to know how to get libraries on board.
The Adafruit CircuitPython Library Bundle

Adafruit provides CircuitPython libraries for much of the hardware they provide, including sensors, breakouts and more. To eliminate the need for searching for each library individually, the libraries are available together in the Adafruit CircuitPython Library Bundle. The bundle contains all the files needed to use each library.

Downloading the Adafruit CircuitPython Library Bundle

You can download the latest Adafruit CircuitPython Library Bundle release by clicking the button below. The libraries are being constantly updated and improved, so you'll always want to download the latest bundle.

Match up the bundle version with the version of CircuitPython you are running. For example, you would download the 6.x library bundle if you're running any version of CircuitPython 6, or the 7.x library bundle if you're running any version of CircuitPython 7, etc. If you mix libraries with major CircuitPython versions, you will get incompatible mpy errors due to changes in library interfaces possible during major version changes.

Download the bundle version that matches your CircuitPython firmware version. If you don't know the version, check the version info in boot_out.txt file on the CIRCUITPY drive, or the initial prompt in the CircuitPython REPL. For example, if you're running v7.0.0, download the 7.x library bundle.

There's also a py bundle which contains the uncompressed python files, you probably don't want that unless you are doing advanced work on libraries.
The CircuitPython Community Library Bundle

The CircuitPython Community Library Bundle is made up of libraries written and provided by members of the CircuitPython community. These libraries are often written when community members encountered hardware not supported in the Adafruit Bundle, or to support a personal project. The authors all chose to submit these libraries to the Community Bundle make them available to the community.

These libraries are maintained by their authors and are not supported by Adafruit. As you would with any library, if you run into problems, feel free to file an issue on the GitHub repo for the library. Bear in mind, though, that most of these libraries are supported by a single person and you should be patient about receiving a response. Remember, these folks are not paid by Adafruit, and are volunteering their personal time when possible to provide support.

Downloading the CircuitPython Community Library Bundle

You can download the latest CircuitPython Community Library Bundle release by clicking the button below. The libraries are being constantly updated and improved, so you'll always want to download the latest bundle.

[Click for the latest CircuitPython Community Library Bundle release](https://adafru.it/VCn)

The link takes you to the latest release of the CircuitPython Community Library Bundle on GitHub. There are multiple versions of the bundle available. Download the bundle version that matches your CircuitPython firmware version. If you don't know the version, check the version info in boot_out.txt file on the CIRCUITPY drive, or the initial prompt in the CircuitPython REPL. For example, if you're running v7.0.0, download the 7.x library bundle.

Understanding the Bundle

After downloading the zip, extract its contents. This is usually done by double clicking on the zip. On Mac OSX, it places the file in the same directory as the zip.
Open the bundle folder. Inside you'll find two information files, and two folders. One folder is the lib bundle, and the other folder is the examples bundle.

Now open the lib folder. When you open the folder, you'll see a large number of .mpy files, and folders.

Example Files

All example files from each library are now included in the bundles in an examples directory (as seen above), as well as an examples-only bundle. These are included for two main reasons:

- Allow for quick testing of devices.
- Provide an example base of code, that is easily built upon for individualized purposes.
Copying Libraries to Your Board

First open the lib folder on your CIRCUITPY drive. Then, open the lib folder you extracted from the downloaded zip. Inside you'll find a number of folders and .mpy files. Find the library you'd like to use, and copy it to the lib folder on CIRCUITPY.

If the library is a directory with multiple .mpy files in it, be sure to copy the entire folder to CIRCUITPY/lib.

This also applies to example files. Open the examples folder you extracted from the downloaded zip, and copy the applicable file to your CIRCUITPY drive. Then, rename it to code.py to run it.

Understanding Which Libraries to Install

You now know how to load libraries on to your CircuitPython-compatible microcontroller board. You may now be wondering, how do you know which libraries you need to install? Unfortunately, it's not always straightforward. Fortunately, there is an obvious place to start, and a relatively simple way to figure out the rest. First up: the best place to start.

When you look at most CircuitPython examples, you'll see they begin with one or more `import` statements. These typically look like the following:

- `import library_or_module`

However, `import` statements can also sometimes look like the following:

- `from library_or_module import name`
They can also have more complicated formats, such as including a `try / except` block, etc.

The important thing to know is that an `import` statement will always include the name of the module or library that you're importing.

Therefore, the best place to start is by reading through the `import` statements.

Here is an example import list for you to work with in this section. There is no setup or other code shown here, as the purpose of this section involves only the import list.

```python
import time
import board
import neopixel
import adafruit_lis3dh
import usb_hid
from adafruit_hid.consumer_control import ConsumerControl
from adafruit_hid.consumer_control_code import ConsumerControlCode
```

Keep in mind, not all imported items are libraries. Some of them are almost always built-in CircuitPython modules. How do you know the difference? Time to visit the REPL.

In the Interacting with the REPL section ([https://adafruit.io/Awz](https://adafruit.io/Awz)) on The REPL page ([http:s://adafruit.io/Awz](http:s://adafruit.io/Awz)) in this guide, the `help("modules")` command is discussed. This command provides a list of all of the built-in modules available in CircuitPython for your board. So, if you connect to the serial console on your board, and enter the REPL, you can run `help("modules")` to see what modules are available for your board. Then, as you read through the `import` statements, you can, for the purposes of figuring out which libraries to load, ignore the statement that import modules.

The following is the list of modules built into CircuitPython for the Feather RP2040. Your list may look similar or be anything down to a significant subset of this list for smaller boards.
Now that you know what you're looking for, it's time to read through the import statements. The first two, `time` and `board`, are on the modules list above, so they're built-in.

The next one, `neopixel`, is not on the module list. That means it's your first library! So, you would head over to the bundle zip you downloaded, and search for neopixel. There is a neopixel.mpy file in the bundle zip. Copy it over to the lib folder on your CIRCUITPY drive. The following one, `adafruit_lis3dh`, is also not on the module list. Follow the same process for adafruit_lis3dh, where you'll find adafruit_lis3dh.mpy, and copy that over.

The fifth one is `usb_hid`, and it is in the modules list, so it is built in. Often all of the built-in modules come first in the import list, but sometimes they don't! Don't assume that everything after the first library is also a library, and verify each import with the modules list to be sure. Otherwise, you'll search the bundle and come up empty!

The final two imports are not as clear. Remember, when `import` statements are formatted like this, the first thing after the `from` is the library name. In this case, the library name is `adafruit_hid`. A search of the bundle will find an adafruit_hid folder. When a library is a folder, you must copy the entire folder and its contents as it is in the bundle to the lib folder on your CIRCUITPY drive. In this case, you would copy the entire adafruit_hid folder to your CIRCUITPY/lib folder.

Notice that there are two imports that begin with `adafruit_hid`. Sometimes you will need to import more than one thing from the same library. Regardless of how many times you import the same library, you only need to load the library by copying over the adafruit_hid folder once.

That is how you can use your example code to figure out what libraries to load on your CircuitPython-compatible board!
There are cases, however, where libraries require other libraries internally. The internally required library is called a dependency. In the event of library dependencies, the easiest way to figure out what other libraries are required is to connect to the serial console and follow along with the `ImportError` printed there. The following is a very simple example of an `ImportError`, but the concept is the same for any missing library.

Example: `ImportError` Due to Missing Library

If you choose to load libraries as you need them, or you're starting fresh with an existing example, you may end up with code that tries to use a library you haven't yet loaded. This section will demonstrate what happens when you try to utilise a library that you don't have loaded on your board, and cover the steps required to resolve the issue.

This demonstration will only return an error if you do not have the required library loaded into the lib folder on your CIRCUITPY drive.

Let's use a modified version of the Blink example.

```python
global led
"""
import board
import time
import simpleio

led = simpleio.DigitalOut(board.LED)

while True:
    led.value = True
    time.sleep(0.5)
    led.value = False
    time.sleep(0.5)
"""
```

Save this file. Nothing happens to your board. Let's check the serial console to see what's going on.

You have an `ImportError`. It says there is no module named 'simpleio'. That's the one you just included in your code!
Click the link above to download the correct bundle. Extract the lib folder from the downloaded bundle file. Scroll down to find simpleio.mpy. This is the library file you're looking for! Follow the steps above to load an individual library file.

The LED starts blinking again! Let's check the serial console.

![Serial Console]

No errors! Excellent. You've successfully resolved an `ImportError`

If you run into this error in the future, follow along with the steps above and choose the library that matches the one you're missing.

**Library Install on Non-Express Boards**

If you have an M0 non-Express board such as Trinket M0, Gemma M0, QT Py M0, or one of the M0 Trinkeys, you'll want to follow the same steps in the example above to install libraries as you need them. Remember, you don't need to wait for an `ImportError` if you know what library you added to your code. Open the library bundle you downloaded, find the library you need, and drag it to the lib folder on your CIRCUITPY drive.

You can still end up running out of space on your M0 non-Express board even if you only load libraries as you need them. There are a number of steps you can use to try to resolve this issue. You'll find suggestions on the [Troubleshooting page](https://adafruit.it/Den).

**Updating CircuitPython Libraries and Examples**

Libraries and examples are updated from time to time, and it's important to update the files you have on your CIRCUITPY drive.

To update a single library or example, follow the same steps above. When you drag the library file to your lib folder, it will ask if you want to replace it. Say yes. That's it!
A new library bundle is released every time there's an update to a library. Updates include things like bug fixes and new features. It's important to check in every so often to see if the libraries you’re using have been updated.

**Code the Protest Sign**

Once your Feather is set up with CircuitPython 5.3.0 or greater, you’ll also need to add some libraries. Follow this [page](https://adafru.it/ABU) for information on how to download and add libraries to your Feather.

From the library bundle you downloaded in that guide page, transfer the following libraries onto the Feather's /lib directory:

- adafruit_bus_device
- adafruit_slideshow.mpy

**Text Editor**

Adafruit recommends using the Mu editor for using your CircuitPython code with the Feather boards. You can get more info in [this guide](https://adafru.it/ANO).

Alternatively, you can use any text editor that saves files.

**Code.py**

Copy the code below and paste it into Mu. Then, save it to your Feather as code.py.

```python
import board
import displayio
import framebufferio
import rgbmatrix
from adafruit_slideshow import SlideShow

displayio.release_displays()
matrix = rgbmatrix.RGBMatrix(
    width=64,
    height=32,
    bit_depth=5,
    addr_pins=[board.A5, board.A4, board.A3, board.A2],
    clock_pin=board.D13,
```

©Adafruit Industries
latch_pin=board.D0,
output_enable_pin=board.D1,
)

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

slideshow = SlideShow( 
display,
backlight_pwm=None,
folder="/images",
loop=True,
order=0,
fade_effect=False,
dwell=8,
auto_advance=True,
)

while slideshow.update():
    pass

The code is quite short! We are using the displayio library along with framebufferio and rgbmatrix to do the heavy lifting.

The adafruit_slideshow library makes it simple to auto-play any images in the specified folder. [Here's more detail](https://adafruit.it/Fx4) on how to use SlideShow.

**Image Files**

You can get started using the included image files, or make your own.

From the project .zip, drag the images folder onto the Feather's CIRCUITPY drive.

The program will automatically use any .bmp files in the /images directory. Make sure they have legal names (no spaces or weird characters!) and are 64x32 pixel .bmp files. 16-bit or 24-bit both work fine.
Now we'll assemble the sign. These steps include:

- Powering the LED panel and Feather/FeatherWing
- Connecting the Feather and FeatherWing to the panel
- 3D printing brackets to attach the diffusion plastic
- Attaching a handle
Power
The panel draws about 2A of current at 5V. The UBEC (universal battery elimination circuit) is a buck converter that will take our roughly ~9.6V of power from the 8 x AA NiMH battery pack and step it down to 5V.

Solder a barrel jack connector so the center leg goes to the positive input wire of the UBEC, and the sleeve leg goes to UBEC ground.

Solder the barrel plug so that the UBEC positive output (red) wire goes to center and the ground to sleeve.

You can then protect the entire circuit with a length of heat shrink tubing as shown here.
Battery Pack
Use Velcro or double-stick foam tape to secure the battery pack, to the sign handle as shown. Make sure to adhere the back side of the battery pack, leaving the lid free to replace batteries as needed.
Add Handle
Measure and mark for two ~3mm holes on the top of the handle. Then drill these out and secure the handle using the two M3 x 12mm screws.
Feather Prep
Solder male header pins under the Feather as shown here. The Feather will stack on top of the RGB Matrix FeatherWing

3D print the Feather protector linked below and then use short M2.5 standoffs and screws to secure the Feather to it.
FeatherWing Prep

Solder female headers to the top of the FeatherWing as shown, and the box header to the underside. This will allow us to stack the Feather on top of the FeatherWing and plug the show shebang into the panel.

Screw the red and black power wires to the terminal block's positive and ground connections respectively. This cable will plug into the power connector on the matrix panel.
Connect to Sign

Plug the FeatherWing into the sign's IDC connector -- make sure to orient it as shown in the photo! (The added resistor and Kapton tape are present due to a reset fix for the gen. 1 board (https://adafruit.it/LHe)).

Then, stack the Feather on top of the FeatherWing.

Connect the battery pack to the UBEC buck converter and then to the power input of the FeatherWing.

Plug the RGB Matrix FeatherWing power output cable into the panel as shown, and then neaten and secure the UBEC and power wiring with a couple of zip ties.
Add Batteries

Throw in a freshly recharged set of NiMH batteries -- these should last about 2-3 hours of constant sign use with typical graphics (full white on all LEDs is the worst case scenario and will deplete the batteries in about an hour), so carry an extra set if you need.
The sign will definitely work in this state, but let's go the extra step and add a sheet of LED diffusion acrylic. This offers the LEDs some protection, and makes the sign much easier to see, especially on video and in pictures. 3D print one set of the brackets linked here to hold the sheet in place.

sign_brackets_all.stl
https://adafruit.it/LHf
Diffusion Sheet
Lay the diffusion plastic sheet on top of the LED panel as shown with the matte side up.

Slide the 3D printed brackets on, then flip the sign over and fasten them in with the M3 x 8mm screws.