Matrix Portal RGB Display Handbag

Created by Erin St Blaine

https://learn.adafruit.com/matrix-portal-rgb-display-handbag

Last updated on 2023-11-13 11:18:47 AM EST
# Table of Contents

**Overview**

**Prep the MatrixPortal**
- Power Prep
- Power Terminals
- Panel Power
- Dual Matrix Setup
- Board Connection

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

**Code the Slideshow**
- Libraries
- Text Editor
- Code
- Pixel Art Specs
- Art Collecting
- How it Works
- Slideshow Setup
- Advance!
- Main Loop

**Build the Window Pocket**
Overview

Make a display for your display! Show off your message to the world. This guide will show how to set up and assemble a Matrix Portal and RGB Matrix display, then make a transparent pocket in your backpack, purse, or messenger bag to show it off.

There is also a simple code example that displays your bmp images in slideshow fashion. Use the onboard buttons to switch between folders and play different messages, or pause on your favorite image. Set your bag down and the onboard accelerometer will sense that you're not moving and dim the screen to save battery life.

You can also use any of the other Matrix Portal code samples to get you started. Get ready to have fun with all the features the Matrix Portal offers.

Difficulty

The build for this project is really easy. For the electronics setup you'll just need a screwdriver. For the handbag mod, we'll use fabric scissors and glue to affix our vinyl window. Sewing skills could be helpful, but are not required.

The coding portion is a little trickier, but I've made it as simple as possible to get images flashing up on your screen. There are more and more guides in the Adafruit Learning System every day with sample code, so this project can get as involved as you'd like.
Parts Needed

The Matrix Portal board is a very cool microcontroller that plugs into the back of any of our RGB Matrix display panels. The displays come in lots of different sizes - check them out here ().

I used the 4mm pitch 64x32 sized one for this project. It measures 10' x 5', and it fits perfectly inside the outer pocket of my pack.

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 Battery Pack - 2200 mAh Capacity - 5V 1A Output
A smaller-sized rechargeable battery pack for your Raspberry Pi or Raspberry...
https://www.adafruit.com/product/1959
This lipstick-sized USB battery will power the Matrix Portal for a few hours (depending on the brightness of your images). We’ve also got a bigger one if you want it to last all night.

![USB Battery Pack for Raspberry Pi - 10000mAh - 2 x 5V outputs](https://www.adafruit.com/product/1566)

A large-sized rechargeable battery pack for your Raspberry Pi (or Arduino, or

This board connects with a USB C cable, so get at least one of these.

<table>
<thead>
<tr>
<th>1 x USB C Cable</th>
<th><a href="https://www.adafruit.com/product/4472">https://www.adafruit.com/product/4472</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>USB C 6&quot; Cable</td>
<td></td>
</tr>
</tbody>
</table>

You can unplug the cable to turn it off, but it's easier and handier to plug in an on/off switch.

<table>
<thead>
<tr>
<th>1 x On/Off Switch</th>
<th><a href="https://www.adafruit.com/product/1620">https://www.adafruit.com/product/1620</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>USB Cable with Switch</td>
<td></td>
</tr>
</tbody>
</table>

Additional Materials

You'll also need a backpack, messenger bag, or purse with a pocket that will comfortably fit your RGB display, with a couple inches extra on the "long" side to accommodate the Matrix Portal board and the USB cable.

I found the pack I'm using in this guide in the travel section at Target.

For the window, you'll need a square of vinyl. You can find this online or at your local plastic store. I used iridescent vinyl, which does make the matrix a little harder to see in the daytime, but it's shiny and iridescent so I don't really care -- and the matrix still shows through beautifully in lower light. The iridescence also gives my images a bit of a rainbow sheen, which I count as a win.

I also used a bit of Fray Check () on my bag. This is a great product that you can find at sewing stores or online. Just apply it to the cut edges of your fabric and it will keep the fabric from unraveling. I always have some of this stuff handy.
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.

Dual Matrix Setup
If you're planning to use a 64x64 matrix, follow these instructions on soldering the Address E Line jumper().
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 () is a derivative of MicroPython () 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

Further Information

For more detailed info on installing CircuitPython, check out Installing CircuitPython ().
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! :)

---

**Code the Slideshow**

This code will display a series of .bmp files in a folder as a slideshow. You can create as many folders as you'd like. The images in each folder will display as their own slideshow.

Press the UP button to pause the slideshow on any image you'd like, and press it again to resume the slideshow.
Press the DOWN button to cycle to the next folder and show the next slideshow.

You can set the amount of time each slide stays on the screen separately for each folder - so you can have a fast scrolling animation for one folder and a slower slideshow for a different folder.

The slideshow will continue to play while you're holding your bag and moving around. If you set your bag down and it stops moving, the Matrix Portal will put the display to sleep after 2 minutes to save battery life.

And of course, you can customize all the timing in the code.

Libraries
We'll need to make sure we have these libraries installed. (Check out this link on installing libraries if needed.)

- adafruit_bus_device
- adafruit_debouncer
- adafruit_lis3dh
- adafruit_matrixportal
- adafruit_slideshow

Text Editor
Adafruit recommends using the Mu editor for editing your CircuitPython code. You can get more info in this guide.

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

Code
Click the Download: Project Zip File link below in the code window to get a zip file with all the files needed for the project. Copy code.py from the zip file and place on the CIRCUITPY drive.
You'll also need to copy the following files to the CIRCUITPY drive. See the graphic at the top of the page as to filenames and where they go):

- /bmps, /bmps2 and /bmps3 directories, which contain the graphics .bmp files for three different sample slide shows.

The button below contains our sample graphics: a VOTE sign, a nyancat animation, and a mermaid slideshow.

bmps.zip

The program will automatically use any .bmp files in these directories. Make sure they have legal names (no spaces or weird characters!) and are a maximum of 64x32 pixels. 16-bit or 24-bit both work fine.

```python
# SPDX-FileCopyrightText: 2020 Melissa LeBlanc-Williams for Adafruit Industries
# SPDX-FileCopyrightText: 2020 Erin St Blaine for Adafruit Industries
#
# SPDX-License-Identifier: MIT

""
# Slideshow Example using the Matrix Portal and 64 x 32 LED matrix display
Written by Melissa LeBlanc-Williams and Erin St Blaine for Adafruit Industries
Images smaller than 64 pixel width will be aligned alternating left or right
Press Up button to pause/resume Slideshow
Press Down button to cycle between slideshow folders
""

import time
import board
import adafruit_lis3dh
import displayio
import busio
from digitalio import DigitalInOut, Pull
from adafruit_matrixportal.matrix import Matrix
from adafruit_slideshow import SlideShow, HorizontalAlignment
from adafruit_debouncer import Debouncer

Display will go to sleep after SLEEP_DURATION seconds have elapsed with no accelerometer movement
Each slide will play for IMAGE_DURATION - customizable for each folder
Add folders to the list to add more slideshows.

SLEEP_DURATION = 60
IMAGE_DURATION = (1, 0.5, 10)
IMAGE_FOLDER = (
    "/bmps",
    "/bmps2",
    "/bmps3",
)

©Adafruit Industries
address=0x19)

_ = ACCEL.acceleration # Dummy reading to blow out any startup residue

pin_down = DigitalInOut(board.BUTTON_DOWN)
pin_down.switch_to_input(pull=Pull.UP)
button_down = Debouncer(pin_down)
pin_up = DigitalInOut(board.BUTTON_UP)
pin_up.switch_to_input(pull=Pull.UP)
button_up = Debouncer(pin_up)

ALIGN_RIGHT = True
auto_advance = True

i = 0
NUM_FOLDERS = 2 # first folder is numbered 0

slideshow = SlideShow(
    display,
    None,
    folder=IMAGE_FOLDER[i],
    order=0,
    auto_advance=False,
    fade_effect=False,
    dwell=IMAGE_DURATION[i],
    h_align=HorizontalAlignment.RIGHT,
)

LAST_ADVANCE = time.monotonic()
last_movement = time.monotonic()
MODE = 1

def advance():
    ''' go to the next slide '''
    # pylint: disable=global-statement
    global ALIGN_RIGHT, LAST_ADVANCE
    ALIGN_RIGHT = not ALIGN_RIGHT
    if ALIGN_RIGHT:
        slideshow.h_align = HorizontalAlignment.RIGHT
    else:
        slideshow.h_align = HorizontalAlignment.LEFT
    LAST_ADVANCE = time.monotonic()
    slideshow.advance()

empty_group = displayio.Group()

while True:
    if ACCEL.shake(shake_threshold=10):
        print("Moving!")
        if MODE == 0:
            slideshow = SlideShow(
                display,
                None,
                folder=IMAGE_FOLDER[i],
                order=0,
                auto_advance=False,
                fade_effect=False,
                dwell=IMAGE_DURATION[i],
                h_align=HorizontalAlignment.RIGHT,
            )
            last_movement = time.monotonic()
        MODE = 1
    elif time.monotonic() > last_movement + SLEEP_DURATION:
        MODE = 0
        display.root_group = empty_group
        if MODE == 1:
            if auto_advance and time.monotonic() > LAST_ADVANCE + IMAGE_DURATION[i]:
                advance()
        button_down.update()
        button_up.update()
if button_up.fell:
    auto_advance = not auto_advance
if button_down.fell:
    i = i + 1
    if i > NUM_FOLDERS:
        i = 0
    slideshow = SlideShow(
        display,
        None,
        folder=IMAGE_FOLDER[i],
        order=0,
        auto_advance=False,
        fade_effect=False,
        dwell=IMAGE_DURATION[i],
        h_align=HorizontalAlignment.RIGHT,
    )

---

**Pixel Art Specs**

If you want to create your own artwork for display, these are the specifications to follow:

- Images should be a maximum of 32 pixels high
- Images can be up to 64 pixels wide
- Colors are 16-bit or 24-bit RGB
- Save files as .bmp format
- We've found that crisp images (not too much anti-aliasing) work best.

We have a whole page on [Pixel Art Fundamentals](#) here!
You can use nearly any paint program, but dedicated pixel art programs work best, such as Aseprite () or the free pixel art app Piskel ()

Art Collecting

If you want to search for ready-made art to display, here are some tips.

Game Frame Art
The excellent Game Frame () product by Jeremy Williams pioneered the pixel art frame. You can check the LedSeq Game Frame forums () for art gallery submissions. Although these tend to be 16x16 pixel images, you can scale them up (don't interpolate!) for use on the Matrix Portal.

Jeremy has also graciously made the code and art for Game Frame available for free download here ().
Search

Another great tip from LedSeq is to use a Google image search to return results of "pixel art" or "video game" at a specific resolution, either 64x32 or 32x32:


Itch.io Assets
Many talented artist post their pixel art game assets to Itch.io () where you can name your price in many case for amazing artwork!

Shown here are the excellent PIPOYA FREE RPG Character Sprites ()

How it Works

Libraries

Here's how the code works. First we import the libraries:

```python
import time
import board
import adafruit_lis3dh
import displayio
import busio
from digitalio import DigitalInOut, Pull
from adafruit_matrixportal.matrix import Matrix
from adafruit_slideshow import SlideShow, HorizontalAlignment
from adafruit_debouncer import Debouncer
```

Variables

Then, we'll set the variables for SLEEP_DURATION (how long until the display goes dark if it's not moving) and IMAGE_DURATION (how long each image stays on the screen in each slide show) in seconds, as well as the path to the image folders.

Add as many folders as you like, but be sure to change the variable for NUM_FOLDERS further down if you add more.
SLEEP_DURATION = 60
IMAGE_DURATION = (1, 0.5, 10)
IMAGE_FOLDER = ("/bmps", "/bmps2", "/bmps3",

Display/Pin Setup

Next, we set up the display and the pins used for the buttons.

```python
matrix = Matrix(bit_depth=6)
display = matrix.display
pin_down = DigitalInOut(board.BUTTON_DOWN)
pin_down.switch_to_input(pull=Pull.UP)
button_down = Debouncer(pin_down)
pin_up = DigitalInOut(board.BUTTON_UP)
pin_up.switch_to_input(pull=Pull.UP)
button_up = Debouncer(pin_up)
```

Slideshow Setup

The alignment of images narrower than 64 pixels wide will alternate, initially with right-side alignment. The `auto_advance` state is set to true initially, but can be changed by pressing the UP button later.

The slideshow is set up next using the settings shown here.

And take a look at the `NUM_FOLDERS` variable -- if you add or remove folders you'll want to change this number. Remember, the first folder is numbered 0, so change this variable to one fewer than you have.

```python
ALIGN_RIGHT = True
auto_advance = True
i = 0
NUM_FOLDERS = 2  # The first folder is numbered 0

slideshow = SlideShow(
    display,
    None,
    "folder=IMAGE_FOLDER[i],
order=0,
auto_advance=False,
fade_effect=False,
dwell=IMAGE_DURATION[i],
    h_align=HorizontalAlignment.RIGHT,
)
LAST_ADVANCE = time.monotonic()
```
Advance!

The `advance()` function will be used to drive the `slideshow.advance()` command using the alternating right/left alignment and auto-advance features.

```python
def advance():
    # pylint: disable=global-statement
global align_right, last_advance
    align_right = not align_right
    if align_right:
        slideshow.h_align = HorizontalAlignment.RIGHT
    else:
        slideshow.h_align = HorizontalAlignment.LEFT
    last_advance = time.monotonic()
    slideshow.advance()
```

Main Loop

Here's what happens during the main loop of the program:

- Check if the display is in motion or not. If so, run the slideshows. If not, put the display to sleep.
- Check if `auto_advance` is on, and if the `time.monotonic()` value is bigger than the `last_advance` time plus the `IMAGE_DURATION` time. If so, `advance()` to the next image!
- Check for button presses:
  - If the UP button is pressed, toggle the auto_advance state
  - If the DOWN button is pressed, manually advance to the next image

```python
while True:
    if ACCEL.shake(shake_threshold=10):
        print("Moving!")
        if MODE == 0:
            slideshow = SlideShow(
                display,
                None,
                folder=IMAGE_FOLDER[i],
                order=0,
                auto_advance=False,
                fade_effect=False,
                dwell=IMAGE_DURATION[i],
                h_align=HorizontalAlignment.RIGHT,
            )
            last_movement = time.monotonic()
            MODE = 1
        elif time.monotonic() > last_movement + SLEEP_DURATION:
            MODE = 0
            display.root_group = empty_group
```
if MODE == 1:
    if auto_advance and time.monotonic() > LAST_ADVANCE + IMAGE_DURATION[i]:
        advance()
    button_down.update()
    button_up.update()
    if button_up.fell:
        auto_advance = not auto_advance
    if button_down.fell:
        i = i + 1
        if i > NUM_FOLDERS:
            i = 0
    slideshow = SlideShow(
        display,
        None,
        folder=IMAGE_FOLDER[i],
        order=0,
        auto_advance=False,
        fade_effect=False,
        dwell=IMAGE_DURATION[i],
        h_align=HorizontalAlignment.RIGHT,
    )

Build the Window Pocket

Cut a piece of vinyl so it's around 1/2" to 1" larger than the face of your RGB matrix on ALL sides (so, 1-2" total extra width).

You can use clear or colored vinyl, or if you're fancy like me, shiny rainbow iridescent unicorn vinyl.
Fold back your pocket flap and mark the width of your RGB matrix with pins. We'll make a slit about 1-2 inches below where the top of the panel will rest.

Use scissors or a utility knife to carefully slice through all the layers of fabric EXCEPT the outermost layer -- leave that layer intact for now. This bag has lots of layers to cut through. For your bag you may only be cutting through one layer of lining.

Slide your assembled display into the pocket you just made and zip the pocket closed. Make sure everything fits with the USB cable connected and the battery attached.

Carefully cut around the face of the panel with sharp fabric scissors.
Depending on your bag's material, you may start to get raveling along the edges. Grab your bottle of Fray Check to nip that in the bud. Apply the Fray Check to any cut edges that look like they might fray -- I put it on the innermost lining and on the outer fabric just to be safe.

Be generous but tidy. We're going to put some trim over this edge later so don't worry if it looks a bit messy.
Place a sheet of scrap paper into your pocket to catch any extra glue drips. Then set the vinyl beneath the outer layer. Add some glue along the seam allowance of the vinyl and carefully press it down.

If you get glue on the vinyl, it's easy to get it off with 99% alcohol and a q-tip/cotton swab. Best to do this fast before the glue dries completely.
Let the glue dry overnight. Come back in the morning and check to be sure all the edges are tightly attached. Once you're happy with the seal, add another layer of E6000 along the outside and glue down your ribbon or trim to hide the raw edge.

Slip your RGB panel inside and see how it fits. My pocket flap is a bit wider than the panel and it was sliding to the side. I want it to stay centered on the window, so I added a couple of stitches with heavy duty thread through the trim, about 2 inches in from each corner. The panel is still easy to remove but now it won't slip to the side.

You can reach inside the top flap to access the control buttons on the top of the Matrix Portal board. If you're using the on/off switch, plug the switch extension between the battery and the USB C cable and you can reach it here too.

Optionally you can edge-finish the raw edges of the pocket with a hand stitch or bias tape, or more sparkly ribbon or trim.