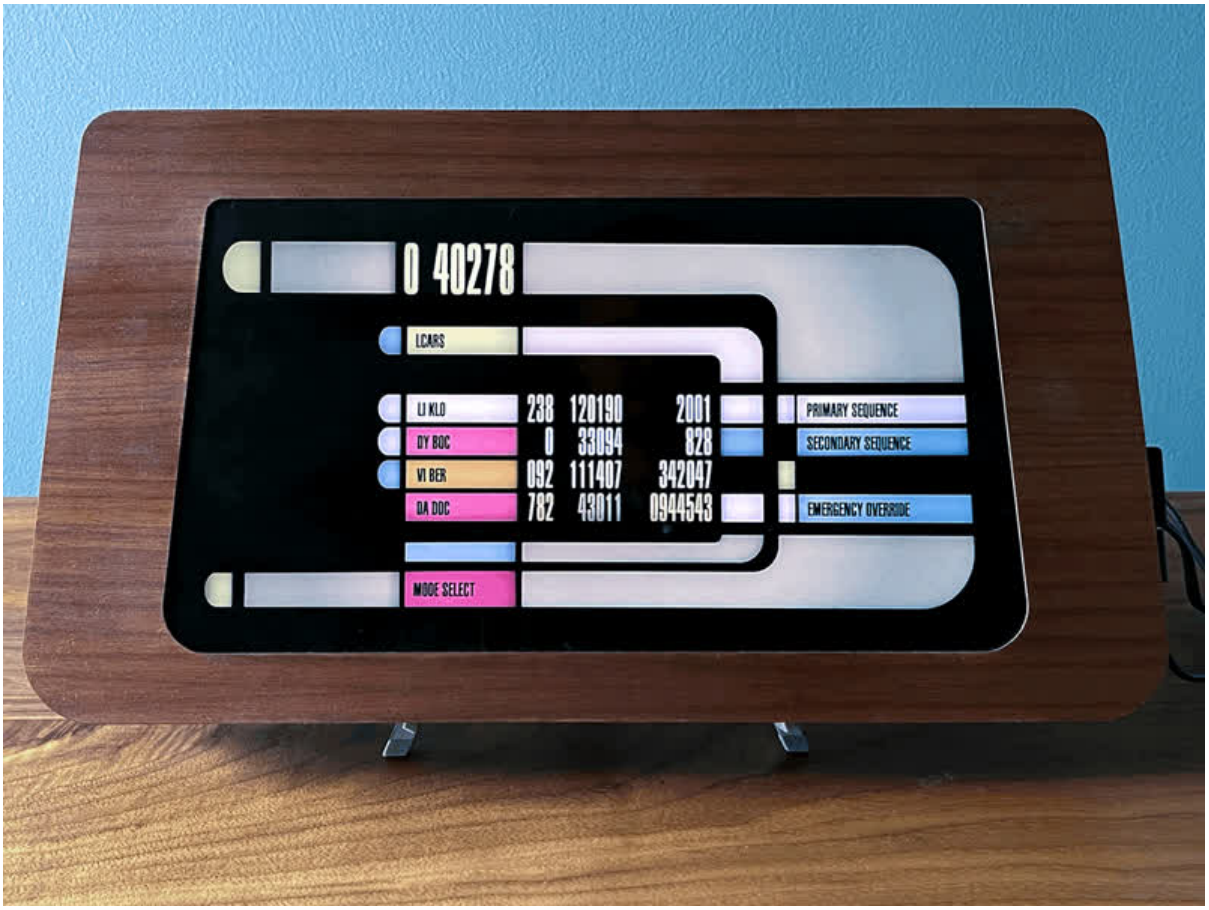
Star Trek LCARS Display

Created by John Park

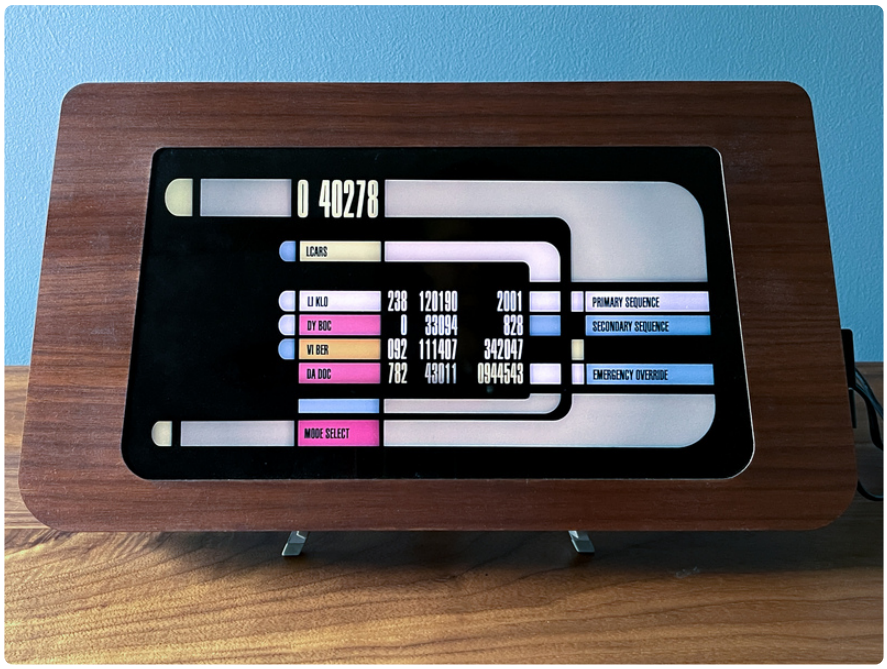


https://learn.adafruit.com/star-trek-lcars-display

Last updated on 2024-04-09 11:15:40 AM EDT

Table of Contents

Overview



My friend worked on the TV series Star Trek: Picard and received this LCARS panel as a gift from the art director. (LCARS is the computer system on board the ships in Star Trek). On it's own the panel doesn't look like much, just a sort of dark piece of acrylic -- it really needs to be backlit, and ideally animated. He asked if I could build a backlit display for him to show it off in his home, and I said "heck yes"!

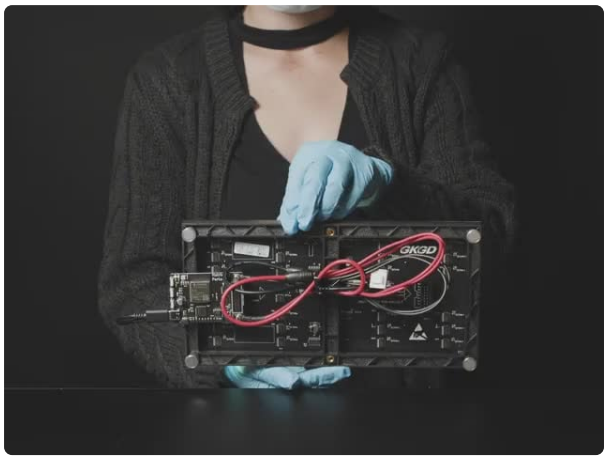
The combination of a light blocker layer and animated sprite sheet make it possible to turn different areas of the display interface "on" and "off", as well as adjust brightness and pattern. This uses the Matrix Portal running CircuitPython to drive a 128 x 64 pixel RGB LED matrix.

Check out the panel on-set in this production photo by art director Liz Kloczkowski (https://adafru.it/18Dw).

For more info on the original LCARS panel construction, check out this excellent blog page (https://adafru.it/18Dx).

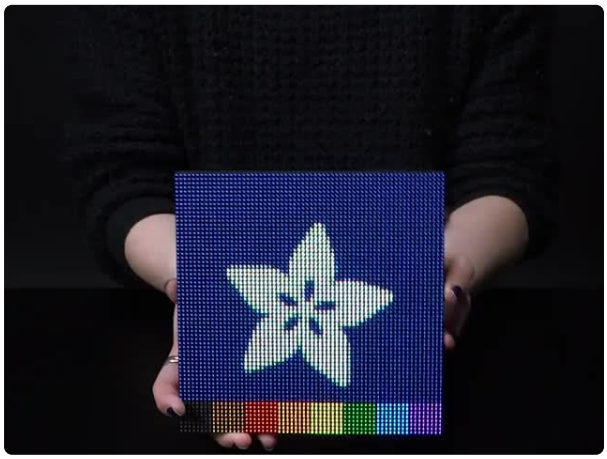
* Backlit film printing can be done at many sign printing shops, you can also create a similar effect using cut vinyl or by painting in reverse on the back side of a piece of glass or acrylic.

Parts



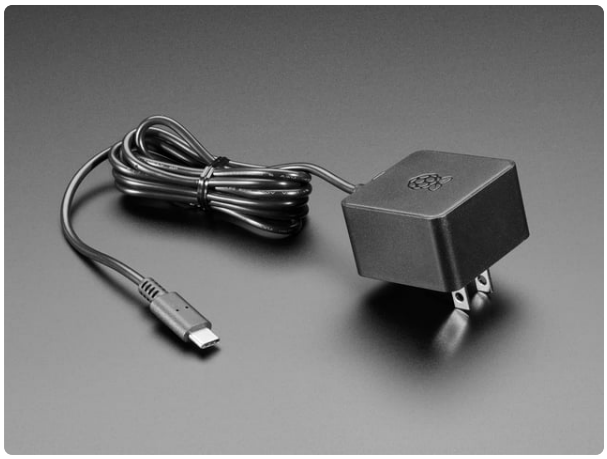
Adafruit Matrix Portal - CircuitPython Powered Internet Display
Folks love our wide selection of RGB matrices and accessories, for making custom colorful LED displays... and our RGB Matrix Shields...
https://www.adafruit.com/product/4745

You can choose any compatible RGB LED matrix display to fit your particular design. The LCARS panel's size and design resolution dictated using **two** of the panels shown below:

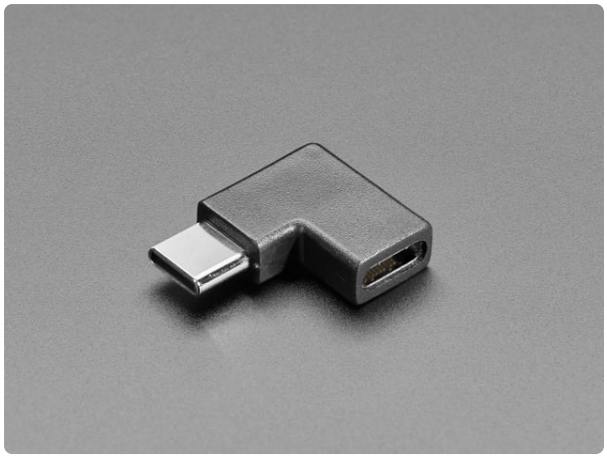


64x64 RGB LED Matrix - 2.5mm Pitch
Winter time can be rough in the city. The sky is gray. The weather is unpredictable. So slough off those seasonal blues with some Times Square razzle dazzle from this...
https://www.adafruit.com/product/3649

You can power a typical display from the USB-C input on the Matrix Portal using a 3A power supply:



Official Raspberry Pi Power Supply 5.1V 3A with USB C
The official Raspberry Pi USB-C power supply is here! And of course, we have 'em in classic Adafruit black! Superfast with just the right amount of cable length to get your Pi 4...
https://www.adafruit.com/product/4298



Right Angle USB Type C Adapter - USB 3.1 Gen 4 Compatible

As technology changes and adapts, so does Adafruit, and speaking of adapting, this right angle adapter is USB C...
https://www.adafruit.com/product/4432

If you are using a pair of denser displays, you'll want to use a separate power supply rated for higher current draw to power the displays:

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...
https://www.adafruit.com/product/1466

Female DC Power adapter - 2.1mm jack to screw terminal block

If you need to connect a DC power wall wart to a board that doesn't have a DC jack - this adapter will come in very handy! There is a 2.1mm DC jack on one end, and a screw terminal...
https://www.adafruit.com/product/368

Black Nylon Machine Screw and Stand-off Set – M3 Thread

Totaling 420 pieces, this M3 Screw Set is a must-have for your workstation. You'll have enough screws, nuts, and hex standoffs to fuel...
https://www.adafruit.com/product/4685

M4 x 35mm socket head screws (https://adafru.it/18Dy) 6ea.

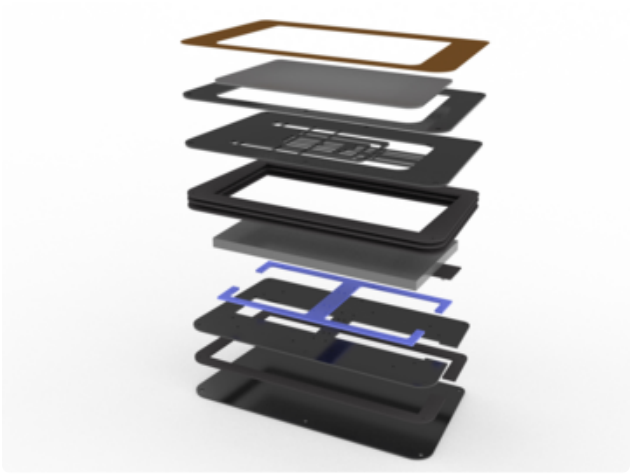
M4 heat set threaded inserts (https://adafru.it/18Dz)6 ea.

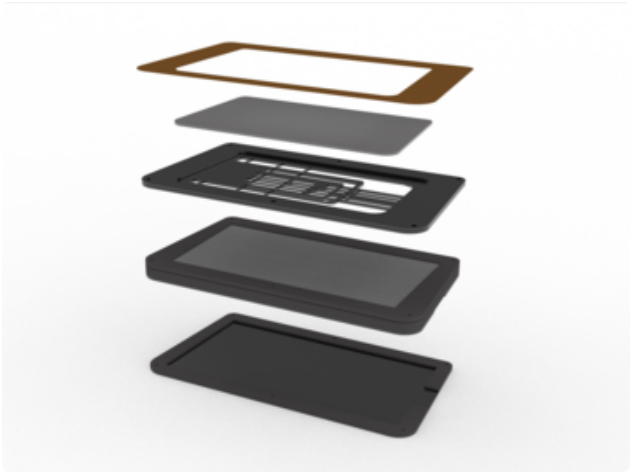
Mahogany veneer (https://adafru.it/18DB) with peel and stick PSA backer adhesive

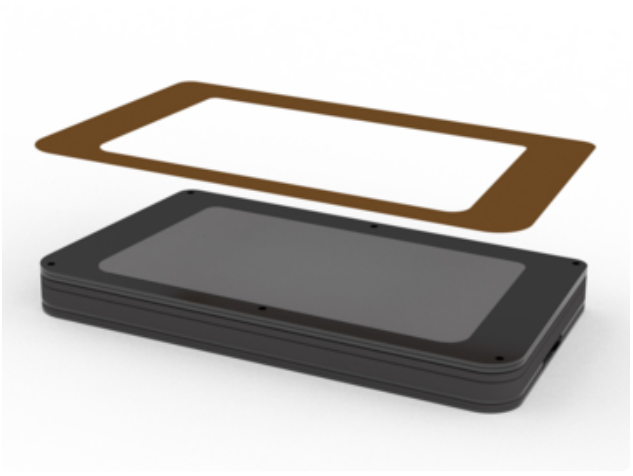
Black acrylic (https://adafru.it/18DD) sheets 12" x 24" 4 ea.

Black 6mm EVA foam (https://adafru.it/18DE) sheet 14" x 36" 2 ea.

Build the LCARS Display

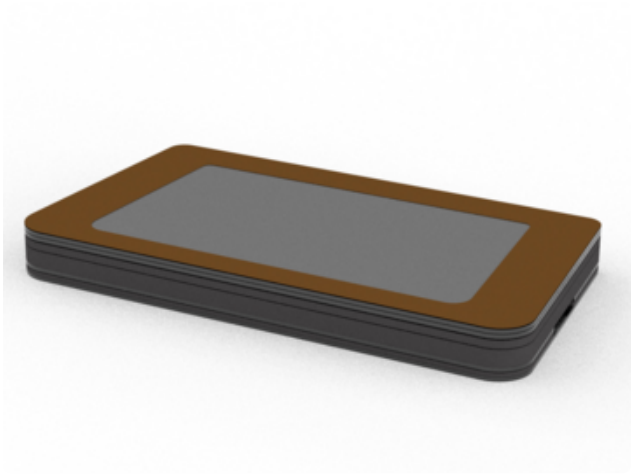






To show off the panel, I built a delicious frame/case sandwich to hold the backlit panel, LED matrices, and the electronics and wiring in place.

The frame is made from acrylic and EVA foam, with a wood veneer on the front panel that is incredibly handsome, matching the wooden details of the Enterprise-D, and functional -- it holds the LCARS panel in place.

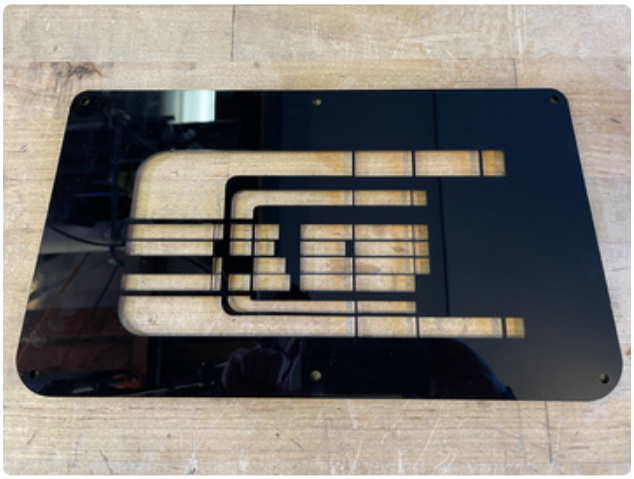


CAD Files

The parts were cut from 3mm acrylic, 6mm EVA foam, and 0.24" (24 mil) wood veneer using an Epilog Zing laser cutter. You can use the attached SVG files to cut your own framing pieces -- if you don't have access to a laser cutter there are online services available. Or, use the files as a printed template to guide using a bandsaw and drill!

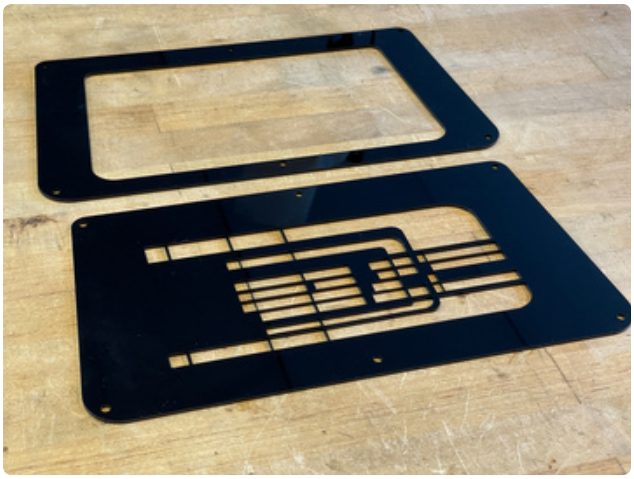
LCARS_frame_CAD.svg

https://adafru.it/18DF



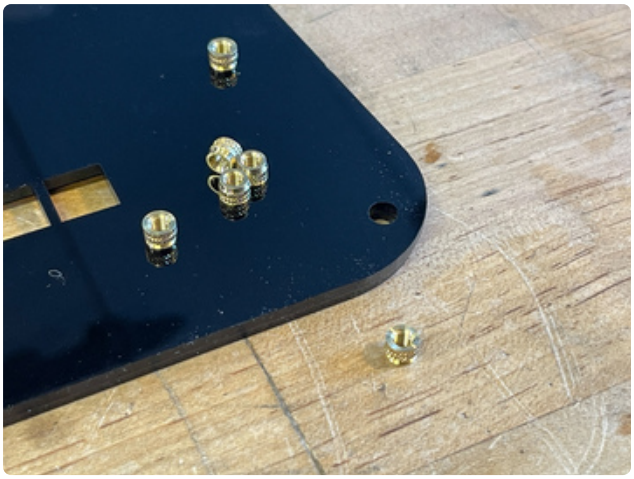
Light Blocker

Use a layer of acrylic cut in a pattern that matches the graphics to avoid any light leakage when segments are turned off. This will help isolate the animation to the intended sections of the backlit graphic.



Front Frame

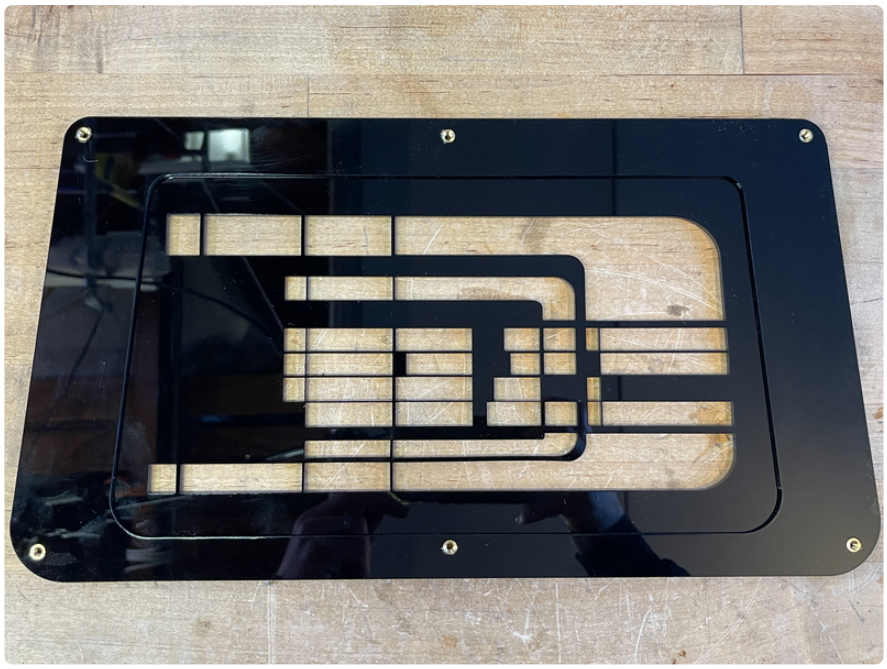
The front frame piece will hold the LCARS panel in place, along with the thin wood veneer placed on top of it.

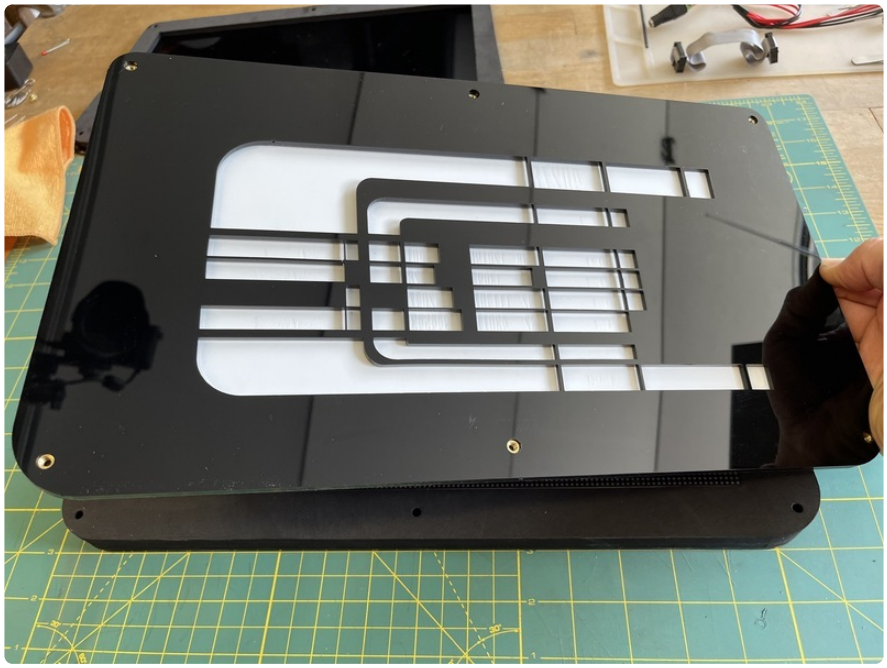


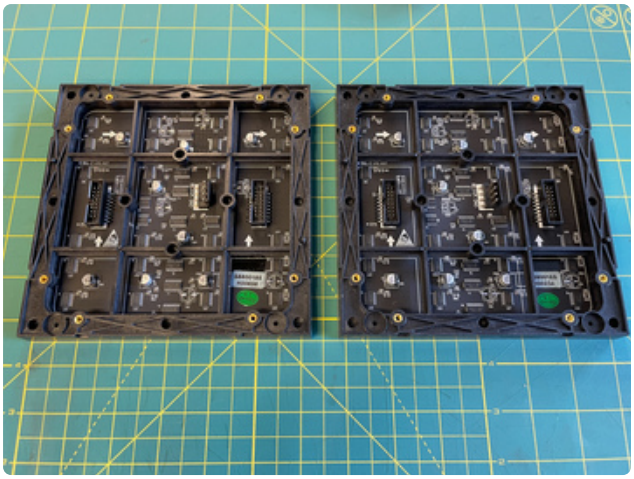
Heat Set Threaded Inserts

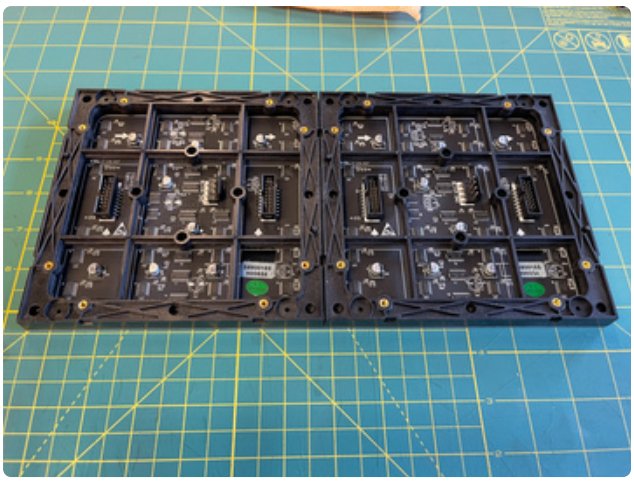
Stack the front frame and then the blocker layers as shown, then insert the threaded inserts using a soldering iron (or specialized tool) as shown. Be sure to the the orientation of the panels correct so they will match the graphic!

Use proper ventilation and air filtration -- don't breath in heated acrylic fumes!



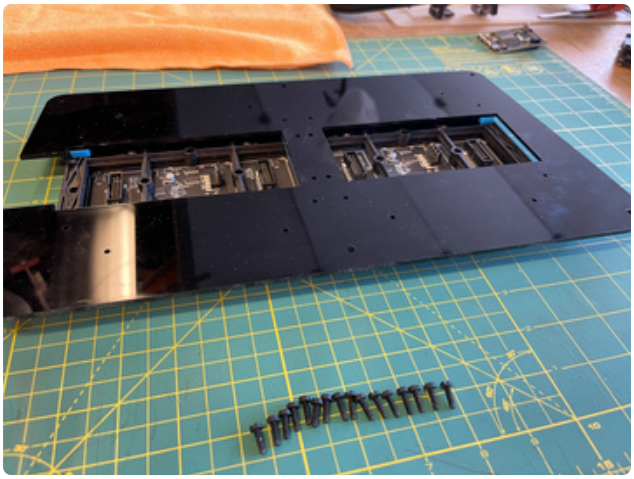


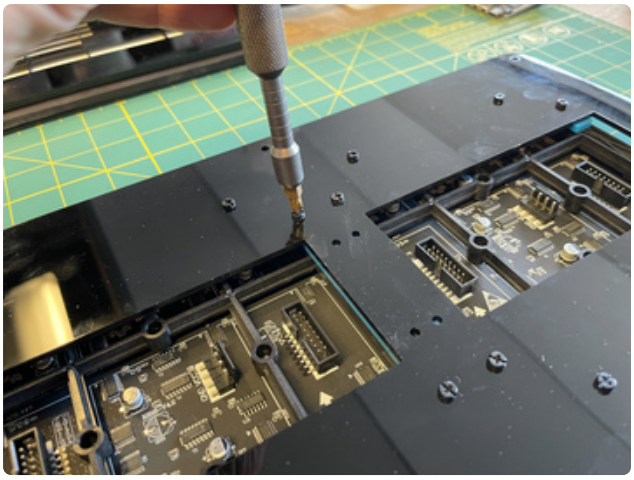




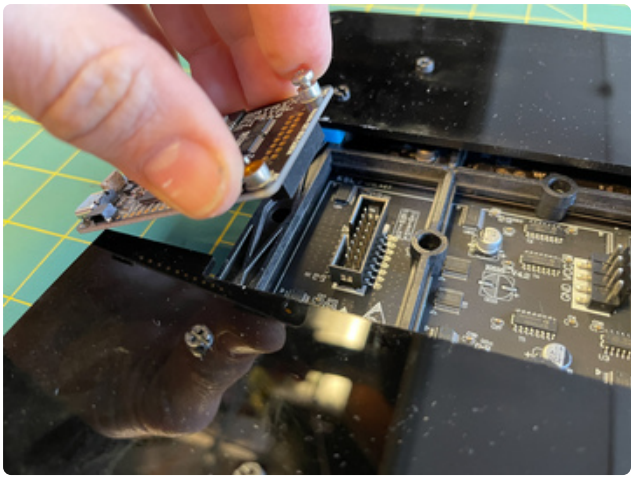
Matrix Bracket

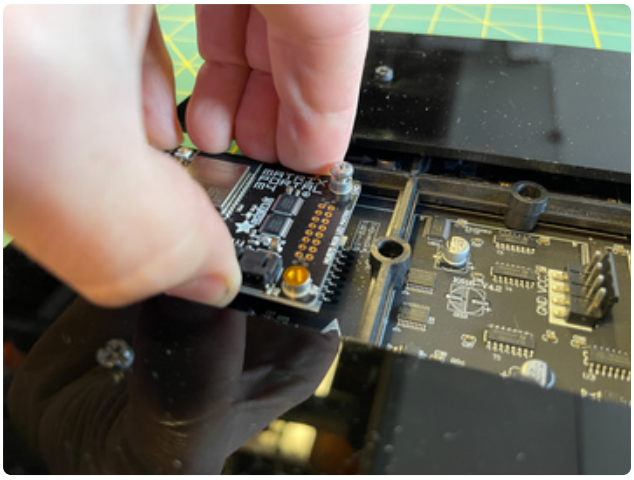
To hold the two LED matrices together, a custom bracket is used.





A second bracket layer is laid over the first and then they are both screwed to the LED panels using M3 screws as shown.

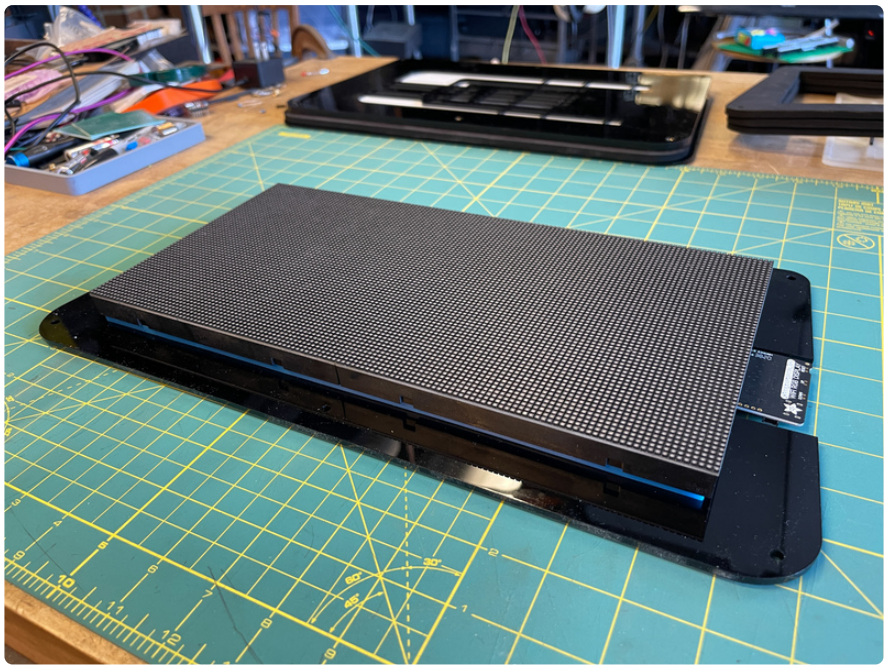


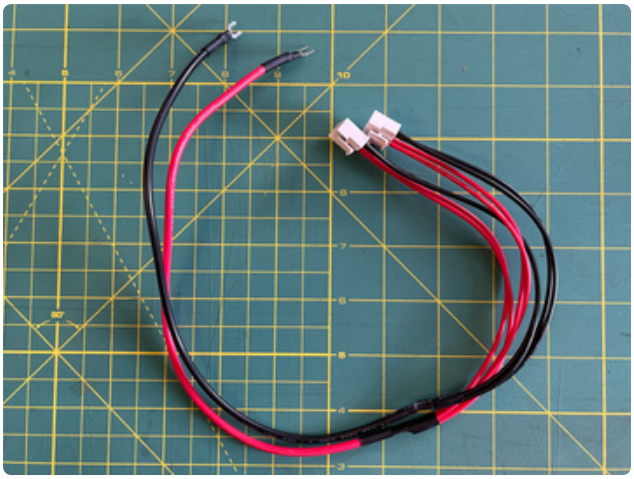


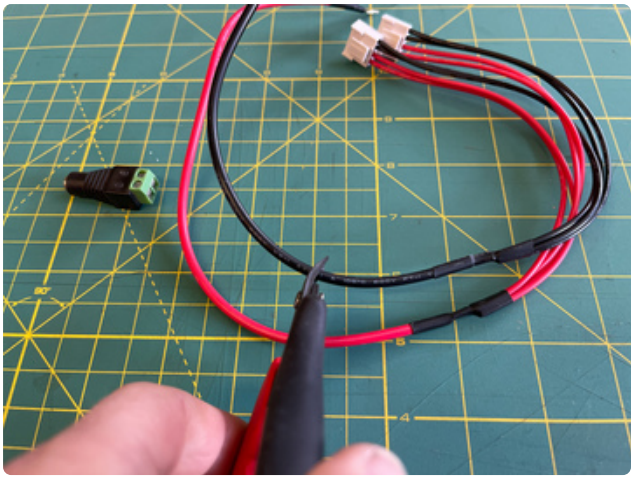
Matrix Portal

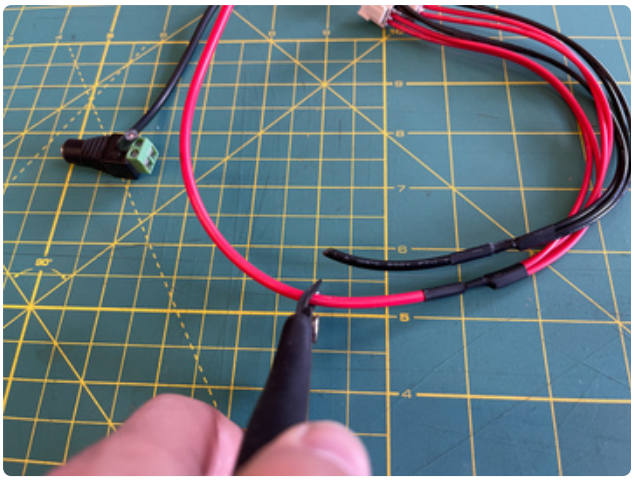
Prep the Matrix Portal by soldering the E-line address jumper to **pin 8** as shown here in the main guide (https://adafru.it/OdJ).

Press the Matrix Portal into the IDC header as shown.





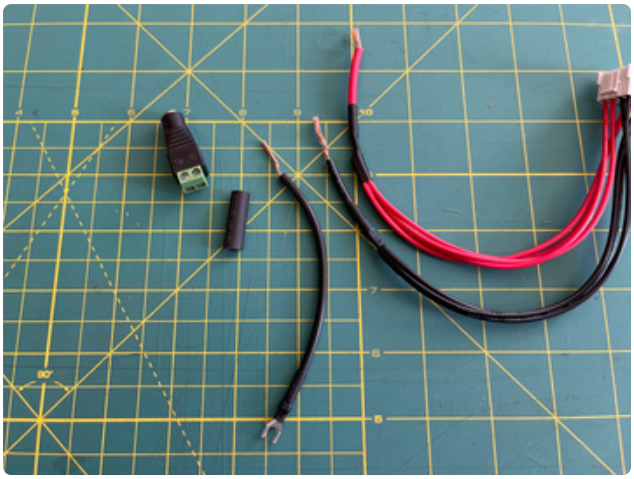


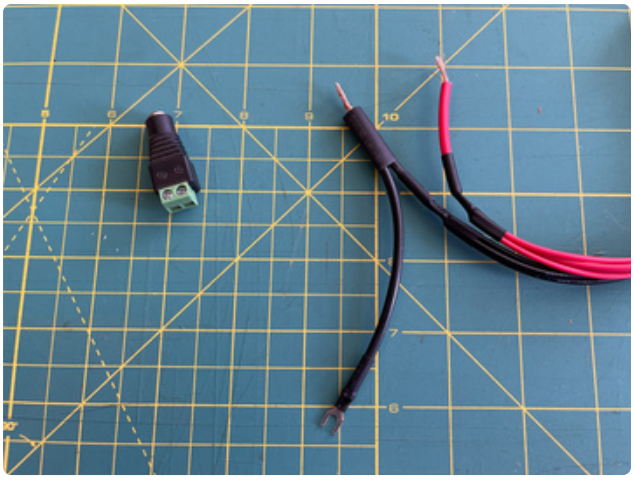


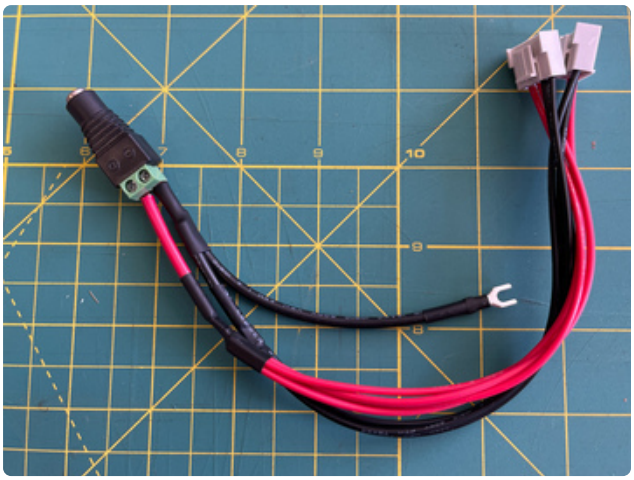
Custom Power Harness

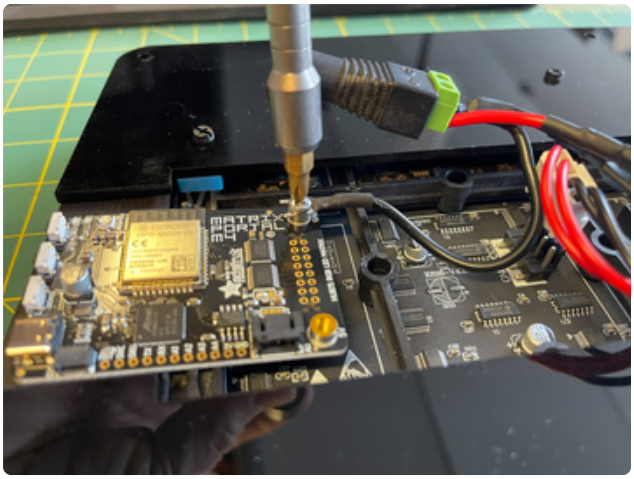
Make a power harness for external power using the included power wiring. Cut the terminal connector ends off and screw the wires into the DC jack adapter

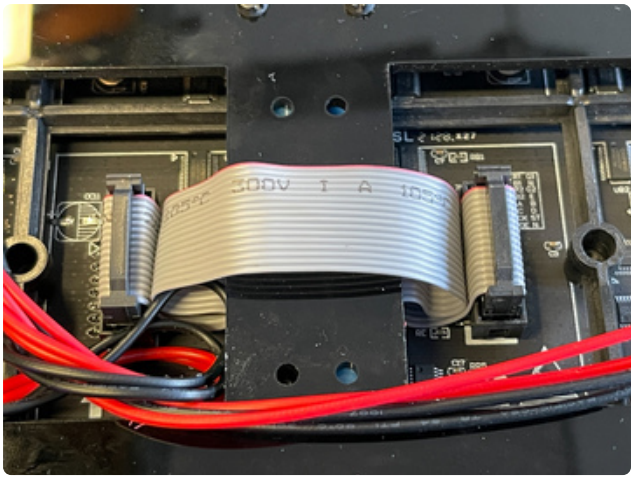
Re-use the black terminal wire end to connect ground to the Matrix Portal's negative (-) terminal screw post.

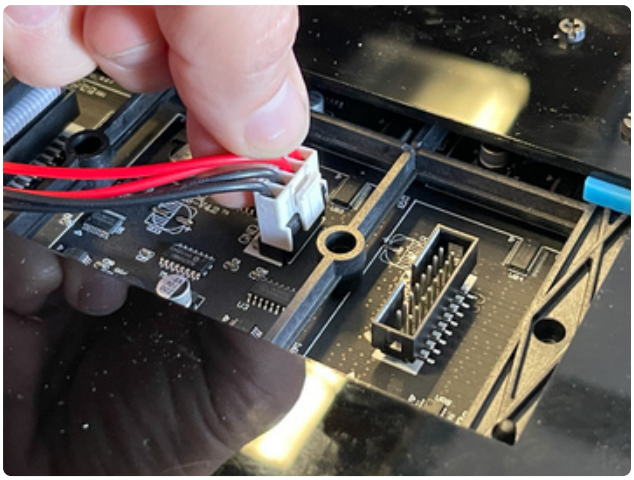


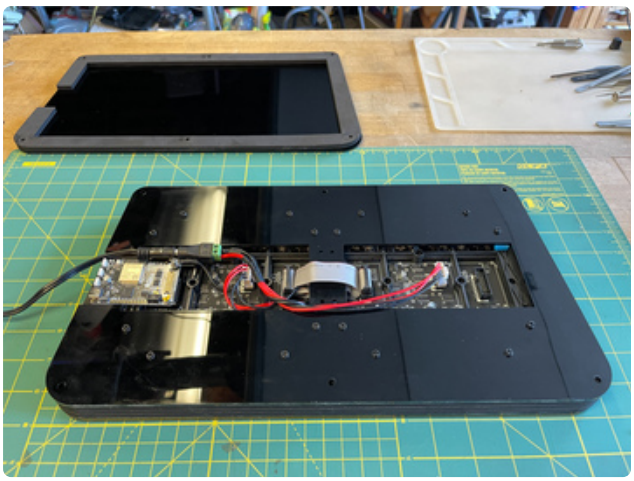








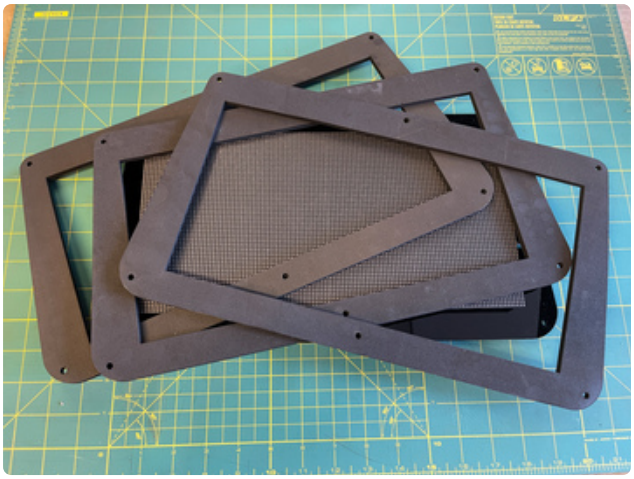


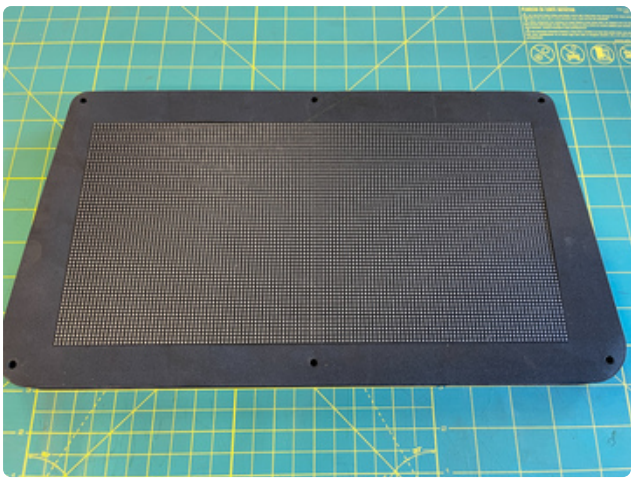


Power and Data

Plug the IDC ribbon cable into both panels to share data.

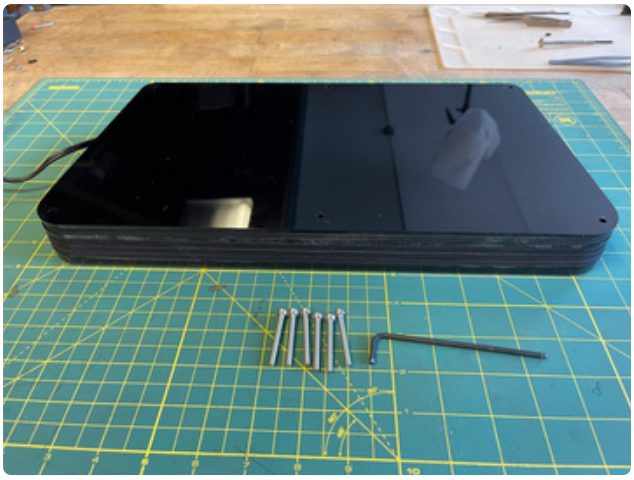
Plug in the power connectors to both panels.

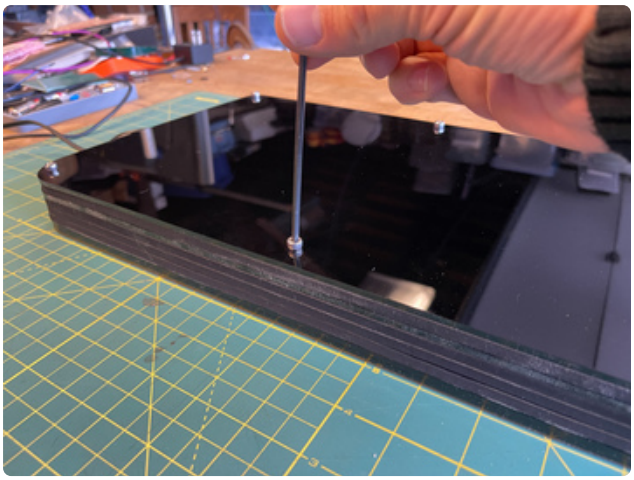


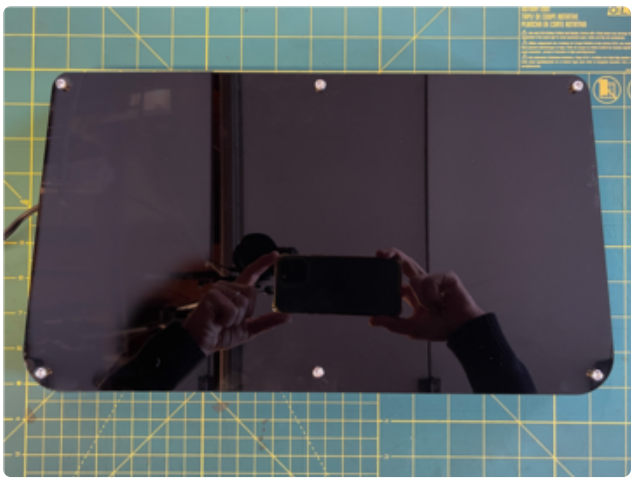


Foam Stack

A stack of three 6mm EVA foam pieces will provide spacing for the front panels, and nestle the LED matrices in place.







Back Panels

Add the back foam spacer (this allows clearance for the power and data cables).

Then, place the back cover on, stack them onto the front panel (don't put the LCARS in yet) and screw these into place with the M4 x 40mm screws.

Veneer Prep

Cut the wood veneer out -- this type of veneer comes with the 3M adhesive backing already applied. I used the laser cutter to the the precise cut, and decided to wait until after cutting to apply stain and finish, since I wasn't eager to test the flammability of the stain and finish combo.

Give the wood a few coats of stain, followed by a spray varnish when dry. Be sure to work in a well ventilated place to avoid fumes.

Adhesive Prep

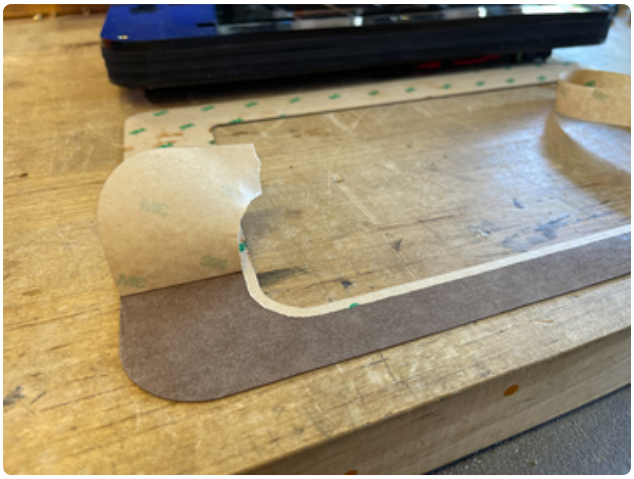
This step is optional -- if you want to avoid getting any adhesive on the backlit panel itself, you can score the peel-back paper in a 4mm offset as shown using tweezers, a compass, a set of calipers, etc.

Veneer Mounting

Place the LCARS into the front frame.

Carefully peel the adhesive backing and place the veneer onto the front panel as shown.

It can help to peel and work the veneer into place from bottom to top. Work carefully, as it can be difficult to remove the veneer once it has adhered to the acrylic.



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!

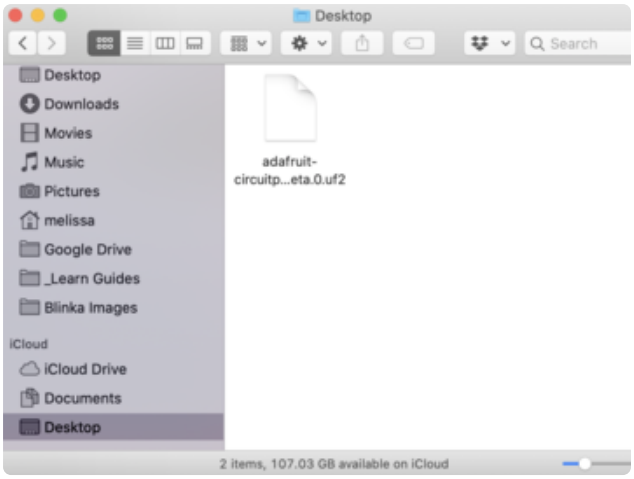
Follow this quick step-by-step for super-fast Python power :)

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

https://adafru.it/Nte

Further Information

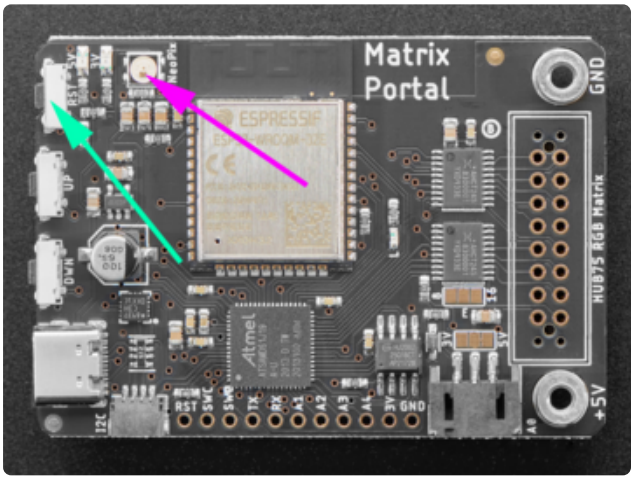
For more detailed info on installing CircuitPython, check out Installing CircuitPython (https://adafru.it/Amd).



Click the link above and download the latest UF2 file.

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

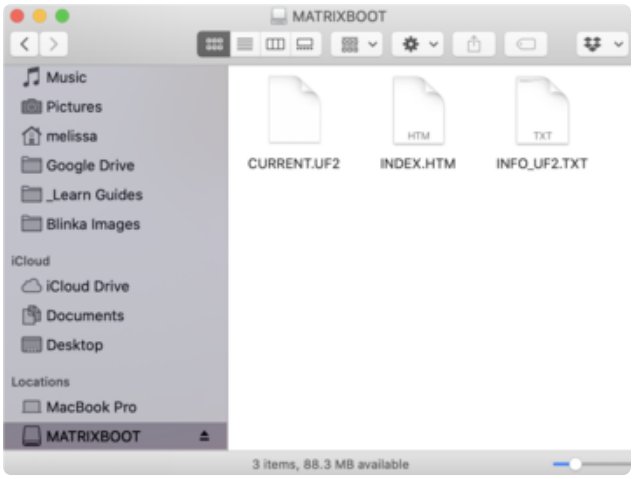
Plug your MatrixPortal M4 into your computer using a known-good USB cable.



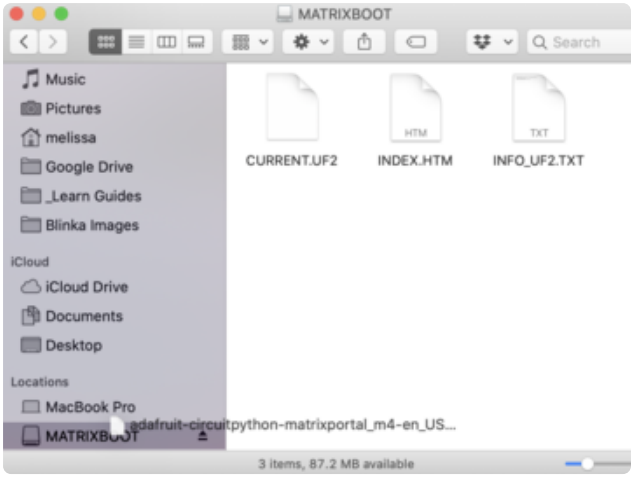
A lot of people end up using charge-only USB cables and it is very frustrating! So make sure you have a USB cable you know is good for data sync.

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.

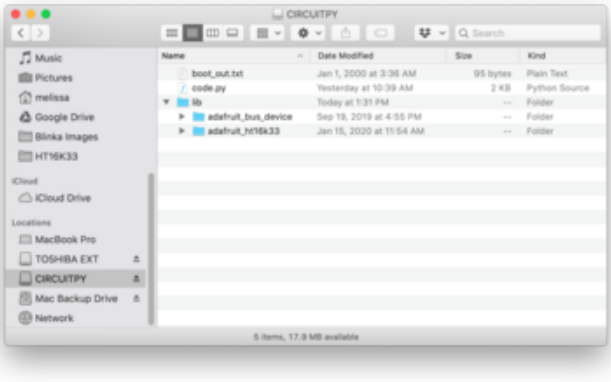
If double-clicking doesn't work the first time, try again. Sometimes it can take a few tries to get the rhythm right!



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! :)

Code the LCARS Display

Text Editor

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

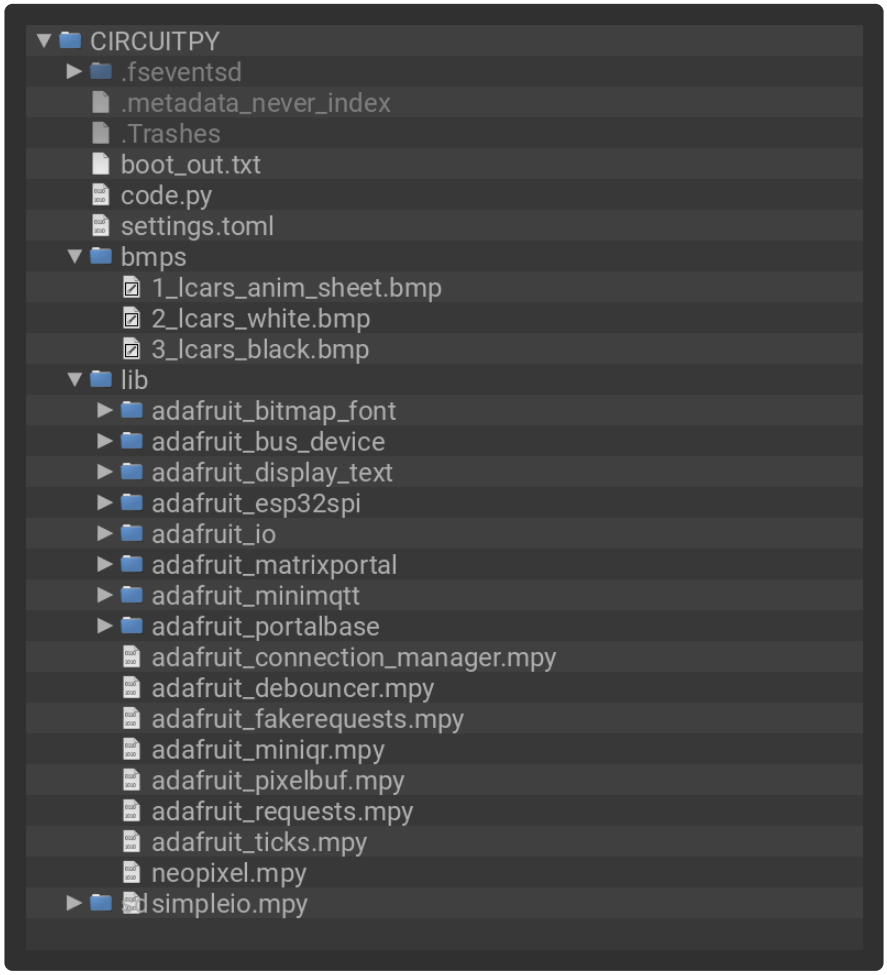
Alternatively, you can use any text editor that saves simple text files.

Download the Project Bundle

Your project will use a specific set of CircuitPython libraries, and the **code.py** file. To get everything you need, click on the **Download Project Bundle** link below, and uncompress the .zip file.

Connect the MatrixPortal to your computer with a known good data+power USB cable. The board should show up in your File Explorer / Finder (depending on your operating system) as a flash drive named **CIRCUITPY**.

Drag the contents of the uncompressed bundle directory onto your board's **CIRCUITPY** drive, replacing any existing files or directories with the same names, and adding any new ones that are necessary.



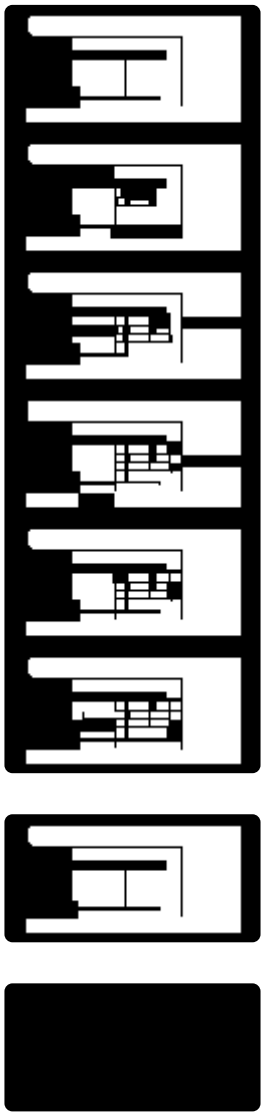

```
# SPDX-FileCopyrightText: 2023 John Park for Adafruit Industries
#
# SPDX-License-Identifier: MIT
# LCARS MatrixPortal Display
# LED brigthness code by Jan Goolsbey
```

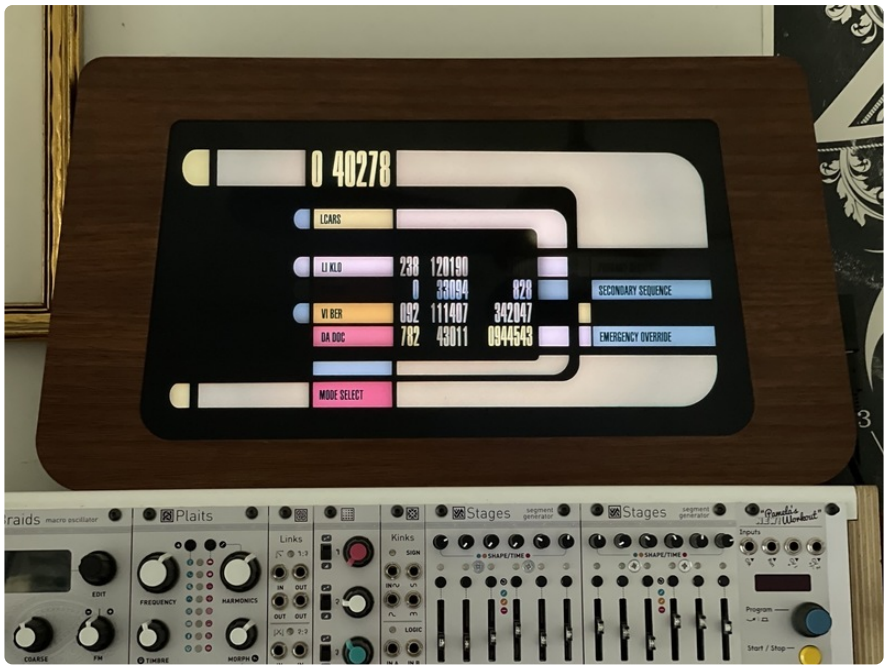
```
import time
import os
import board
import displayio
from digitalio import DigitalInOut, Pull
from simpleio import map_range  # For color brightness calculation
from adafruit_matrixportal.matrix import Matrix
from adafruit_debouncer import Debouncer

import supervisor
supervisor.runtime.autoreload = True

SPRITESHEET_FOLDER = "/bmps"
DEFAULT_FRAME_DURATION = 0.7  # 100ms
AUTO_ADVANCE_LOOPS = 3
bitmap = ""
brightness = 15  # ### Integer value from 0 to 15

# --- Display setup ---
matrix = Matrix(bit_depth=4, width=128, height=64)
sprite_group = displayio.Group()
matrix.display.root_group = sprite_group

# --- Button setup ---
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)

auto_advance = False

file_list = sorted(
    [
        f
        for f in os.listdir(SPRITESHEET_FOLDER)
        if (f.endswith(".bmp") and not f.startswith("."))
    ]
)

if len(file_list) == 0:
    raise RuntimeError("No images found")

current_image = None
current_frame = 0
current_loop = 0
frame_count = 0
frame_duration = DEFAULT_FRAME_DURATION

def image_brightness(new_bright=0):
    """Calculate the white color brightness.
    Returns a white RBG888 color value proportional to `new_bright`."""
    # Scale brightness value
    bright = int(map_range(new_bright, 0, 15, 0x00, 0xFF))
    # Recombine and return a composite RGB888 value
    return (bright << 16) + (bright << 8) + bright

def load_image():
    """
    Load an image as a sprite
    """
    # pylint: disable=global-statement
    global current_frame, current_loop, frame_count, frame_duration, bitmap
    while sprite_group:
        sprite_group.pop()

    filename = SPRITESHEET_FOLDER + "/" + file_list[current_image]
```

```
    bitmap = displayio.OnDiskBitmap(filename)
    ### Change the palette value proportional to BRIGHTNESS
    bitmap.pixel_shader[1] = image_brightness(brightness)
    sprite = displayio.TileGrid(
        bitmap,
        pixel_shader=bitmap.pixel_shader,
        tile_width=bitmap.width,
        tile_height=matrix.display.height,
    )

    sprite_group.append(sprite)

    current_frame = 0
    current_loop = 0
    frame_count = int(bitmap.height / matrix.display.height)
    frame_duration = DEFAULT_FRAME_DURATION


def advance_image():
    """
    Advance to the next image in the list and loop back at the end
    """
    # pylint: disable=global-statement
    global current_image
    if current_image is not None:
        current_image += 1
    if current_image is None or current_image >= len(file_list):
        current_image = 0
    load_image()


def advance_frame():
    """
    Advance to the next frame and loop back at the end
    """
    # pylint: disable=global-statement
    global current_frame, current_loop
    current_frame = current_frame + 1
    if current_frame >= frame_count:
        current_frame = 0
        current_loop = current_loop + 1
    sprite_group[0][0] = current_frame

advance_image()

last_time = time.monotonic()


while True:
    button_down.update()
    button_up.update()
    if button_up.fell:
        advance_image()
    if button_down.fell:
        brightness = (brightness + 2) % 16
        print(brightness)
        bitmap.pixel_shader[1] = image_brightness(brightness)  # ### Change the
brightness

    if time.monotonic() - last_time > frame_duration:
        advance_frame()
        last_time = time.monotonic()
```

Bitmaps

Included are the following 4-bit 128x64 pixel bitmap files. One is blank (for "off"), one has white pixels that align with the lit LCARS segments, and the third one is a vertical sprite sheet that animates the segments.





How it Works

Libraries

First, to import the `time, os, board, displayio, digitalio, simpleio, adafruit_matrixportal,` and `adafruit_debouncer` libraries.

```
import time
import os
import board
import displayio
from digitalio import DigitalInOut, Pull
from simpleio import map_range  # For color brightness calculation
from adafruit_matrixportal.matrix import Matrix
from adafruit_debouncer import Debouncer
```

Variables

Next, to set up variables for the sprite playback.

```
SPRITESHEET_FOLDER = "/bmps"
DEFAULT_FRAME_DURATION = 0.7  # 100ms
AUTO_ADVANCE_LOOPS = 3
brightness = 15
```

Display Setup

The display setup instantiates the matrix display with the proper bit depth, width and height settings.

```
# --- Display setup ---
matrix = Matrix(bit_depth=4, width=128, height=64)
sprite_group = displayio.Group()
matrix.display.root_group = sprite_group
```

Button Setup

Use the two user buttons on the Matrix Portal to adjust the brightness of the panel and to pick different sprites (still, animated, blank).

```
# --- Button setup ---
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)
```

File List

This code is used to check for the images inside of the ".bmp" folder on the **CIRCUITPY** drive.

```
file_list = sorted(
    [
        f
        for f in os.listdir(SPRITESHEET_FOLDER)
        if (f.endswith(".bmp") and not f.startswith("."))
    ]
)

if len(file_list) == 0:
    raise RuntimeError("No images found")

current_image = None
current_frame = 0
current_loop = 0
frame_count = 0
frame_duration = DEFAULT_FRAME_DURATION
```

Image Brightness Function

This handy function (created by Jan Goolsbey, thanks Jan!) is used to adjust the brightness value of the RGB LED matrix.

```
def image_brightness(new_bright=0):
    """Calculate the white color brightness.
    Returns a white RBG888 color value proportional to `new_bright`."""
    # Scale brightness value
    bright = int(map_range(new_bright, 0, 15, 0x00, 0xFF))
    # Recombine and return a composite RGB888 value
    return (bright << 16) + (bright << 8) + bright
```

Load Image, Advance Image and Advance Frame Functions

These functions are used to load images as sprites and advance between them (with a button press later).

The `advance_frame()` function is used to move among the frames of a sprite sheet for animation.

```
def load_image():
    """
    Load an image as a sprite
    """
    # pylint: disable=global-statement
    global current_frame, current_loop, frame_count, frame_duration, bitmap
    while sprite_group:
        sprite_group.pop()

    filename = SPRITESHEET_FOLDER + "/" + file_list[current_image]

    bitmap = displayio.OnDiskBitmap(filename)
    ### Change the palette value proportional to BRIGHTNESS
    bitmap.pixel_shader[1] = image_brightness(brightness)
    sprite = displayio.TileGrid(
        bitmap,
        pixel_shader=bitmap.pixel_shader,
        tile_width=bitmap.width,
        tile_height=matrix.display.height,
    )

    sprite_group.append(sprite)

    current_frame = 0
    current_loop = 0
    frame_count = int(bitmap.height / matrix.display.height)
    frame_duration = DEFAULT_FRAME_DURATION

def advance_image():
    """
    Advance to the next image in the list and loop back at the end
    """
    # pylint: disable=global-statement
    global current_image
    if current_image is not None:
        current_image += 1
    if current_image is None or current_image >= len(file_list):
        current_image = 0
    load_image()


def advance_frame():
    """
    Advance to the next frame and loop back at the end
    """
    # pylint: disable=global-statement
    global current_frame, current_loop
    current_frame = current_frame + 1
    if current_frame >= frame_count:
        current_frame = 0
        current_loop = current_loop + 1
    sprite_group[0][0] = current_frame
```

Main Loop

The main loop of the program checks for button presses and then calls a function.

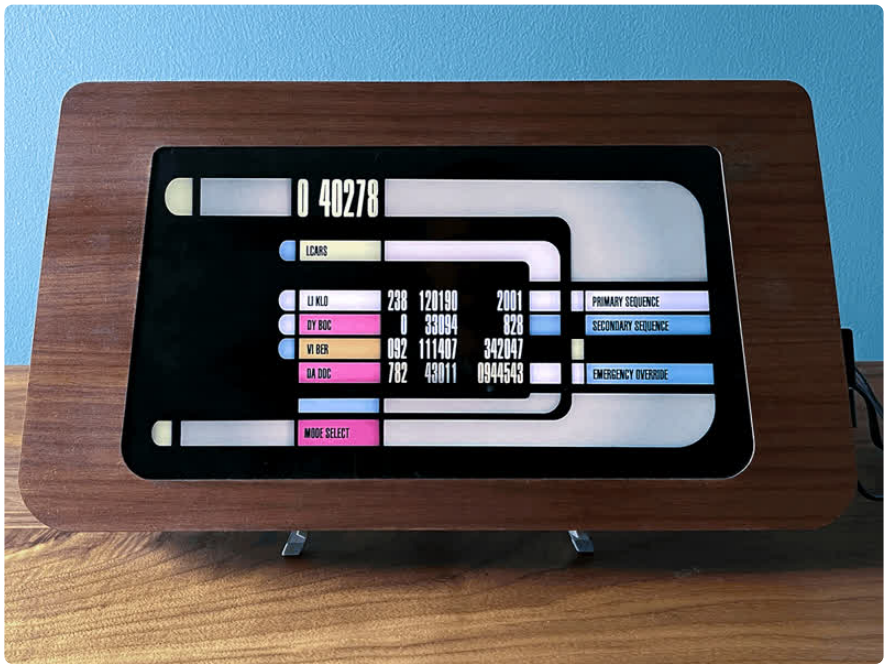
When the down button is pressed, the image brightness increases (each click increases brightness until we loop back around to 0 on the eighth click).

When the up button is pressed, the `advance_image()` function is called, and the next image in the .bmp folder is displayed.

When the current `time.monotonic()` minus the `last_time` is greater than the `frame_duration`, the next frame of the sprite sheet is displayed.

```
while True:
    button_down.update()
    button_up.update()
    if button_up.fell:
        advance_image()
    if button_down.fell:
        brightness = (brightness + 2) % 16
        print(brightness)
        bitmap.pixel_shader[1] = image_brightness(brightness)  # ### Change the
brightness

    if time.monotonic() - last_time > frame_duration:
        advance_frame()
        last_time = time.monotonic()
```

Use the LCARS



To use the display, plug in both the USB-C power and the 5V 4A power. The display will start up automatically.

Control the image using the UP button on the Matrix Portal (not the topmost button, which is the RESET button). This will cycle through any images in the .bmp directory. One black image has been included to turn the display "off".

The Matrix Portal's DOWN button is used to control levels of brightness. There are eight levels, from off to full brightness, which cycle each time the button is pressed.