# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>3</td>
</tr>
<tr>
<td>• Parts</td>
<td></td>
</tr>
<tr>
<td>Assembly</td>
<td>6</td>
</tr>
<tr>
<td>Prep the MatrixPortal</td>
<td>6</td>
</tr>
<tr>
<td>• Power Prep</td>
<td></td>
</tr>
<tr>
<td>• Power Terminals</td>
<td></td>
</tr>
<tr>
<td>• Panel Power</td>
<td></td>
</tr>
<tr>
<td>• Dual Matrix Setup</td>
<td></td>
</tr>
<tr>
<td>• Board Connection</td>
<td></td>
</tr>
<tr>
<td>LED Matrix Diffuser</td>
<td>10</td>
</tr>
<tr>
<td>• LED Diffusion Acrylic</td>
<td></td>
</tr>
<tr>
<td>• Measure and Cut the Plastic</td>
<td></td>
</tr>
<tr>
<td>• Uglu Dashes</td>
<td></td>
</tr>
<tr>
<td>• Stand</td>
<td></td>
</tr>
<tr>
<td>Adafruit IO</td>
<td>16</td>
</tr>
<tr>
<td>• What can Adafruit IO do for me?</td>
<td></td>
</tr>
<tr>
<td>Get Started with Adafruit IO</td>
<td>17</td>
</tr>
<tr>
<td>• I have an Adafruit.com Account already</td>
<td></td>
</tr>
<tr>
<td>• Create an Adafruit Account (for Adafruit IO)</td>
<td></td>
</tr>
<tr>
<td>Upgrade to Adafruit IO Plus</td>
<td>20</td>
</tr>
<tr>
<td>• Subscribe to Adafruit IO Plus</td>
<td></td>
</tr>
<tr>
<td>• IO Plus FAQ</td>
<td></td>
</tr>
<tr>
<td>Enable SMS</td>
<td>22</td>
</tr>
<tr>
<td>Create Feeds</td>
<td>23</td>
</tr>
<tr>
<td>Matrix Portal Code</td>
<td>26</td>
</tr>
<tr>
<td>Install CircuitPython</td>
<td>27</td>
</tr>
<tr>
<td>• Set up CircuitPython Quick Start!</td>
<td></td>
</tr>
<tr>
<td>• Further Information</td>
<td></td>
</tr>
<tr>
<td>Connect to WiFi</td>
<td>29</td>
</tr>
<tr>
<td>• CircuitPython settings.toml File</td>
<td></td>
</tr>
<tr>
<td>• settings.toml File Tips</td>
<td></td>
</tr>
<tr>
<td>• Accessing Your settings.toml Information in code.py</td>
<td></td>
</tr>
<tr>
<td>Code the Message Board</td>
<td>32</td>
</tr>
<tr>
<td>• Text Editor</td>
<td></td>
</tr>
<tr>
<td>• Upload the Code, Fonts, Images and Libraries to the MatrixPortal S3</td>
<td></td>
</tr>
<tr>
<td>• Customize your Code</td>
<td></td>
</tr>
<tr>
<td>• Feed Settings</td>
<td></td>
</tr>
<tr>
<td>• Font Setup</td>
<td></td>
</tr>
<tr>
<td>• Going Deeper</td>
<td></td>
</tr>
</tbody>
</table>
Overview

Adafruit's RGB Matrix Panels put flashy, custom signage within reach of everyone. Create your own custom LED sign and add text, images, or video. The plug-and-play Matrix Portal S3 board also adds web connectivity for live data feeds or internet-of-things capabilities. You can also add sensors for interactivity. These boards have so much potential that it can almost get overwhelming.

Now it's easier than ever to add content to your RGB Matrix Panel using a text messaging app on your phone.

Set up a smart phone with our Adafruit IO+ service, and send text messages that magically scroll across your matrix in bright colors. Update your sign from anywhere, or give the control to your employees or family members, since there's no coding or computer needed to update your sign.

Difficulty

This is a medium-difficulty project. There is no soldering involved - the pieces all fit together with plug-in connectors and screw terminals, so the only tool you need for assembly is a screwdriver.

For security reasons, the text messages need to come from one particular phone number. Once the setup is done, just send a text from that phone to Adafruit IO+, which then sends the message on to your Matrix Portal S3 via WiFi. The message you sent will scroll across the screen in your chosen font and color.
Costs

For this project you will need to subscribe to Adafruit IO+, which costs $10/month or $99 per year, and allows you to send up to 25 SMS messages per day.

Steps

1. Assembly: Plug your Matrix Portal S3 into your RGB Panel and connect the power wires
2. Adafruit IO+: Set up your account at Adafruit IO+, our Internet-of-Things server, connect your phone number, and set up feeds
3. CircuitPython Code: Install CircuitPython on your Matrix Portal S3 and code the message board

We carry matrix panels in a variety of sizes and resolutions. Take a look at Graphic LEDs() in our store for all the options.

Parts

Adafruit Matrix Portal S3 CircuitPython Powered Internet Display
Folks love our wide selection of RGB matrices and accessories for making custom colorful LED displays... and our RGB Matrix...
https://www.adafruit.com/product/5778

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
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

Medium 16x32 RGB LED matrix panel - 6mm Pitch
Bring a little bit of Times Square into your home with this 16 x 32 RGB LED matrix panel. These panels are normally used to make video walls, here in New York we see them on the sides...
https://www.adafruit.com/product/420

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

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
Assembly

Putting your RGB Matrix Panel together is very simple. The Matrix Portal S3 plugs directly into the back of the board, and the only tool you need is a screwdriver.

For most Matrix Panels you can power directly off the Matrix Portal's USB port. If you are chaining panels together and need more power, check out this guide for wiring options.

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.
You can add an LED diffusion acrylic faceplate to your LED matrix display. (Pictured here with the ON AIR project)

This can help protect the LEDs as well as enhance the look of the sign both indoors and out by reducing glare and specular highlights of the plastic matrix grid.
Measure and Cut the Plastic

You can use the sign to measure and mark cut lines on the paper backing of the acrylic sheet.

Then, use a tablesaw or bandsaw with a fine toothed blade and a guide or sled to make the cuts.

Note: it is possible to score and snap acrylic, but it can be very tricky to get an even snap without proper clamping.
Peel away the paper backing from both sides and set the acrylic onto your matrix display.
The best method we've found for adhering acrylic to the matrix display is to use Uglu Dashes clear adhesive rectangles from Pro Tapes. They are incredibly strong (although can be removed if necessary), easy to apply, and are invisible once attached.

Use one at each corner and one each at the halfway point of the long edges, then press the acrylic and matrix panel together for about 20 seconds.

Here you can see the impact of using the diffusion acrylic. (Pictured here with the ON AIR sign project)
Stand
A very simple and attractive way to display your matrix is with the adjustable bent-wire stand ().
Alternately, you can use a frame, 3D printed brackets, tape, glue, or even large binder clips to secure the acrylic to the sign and then mount it on a wall, shelf, or display cabinet.

These mini-magnet feet can be used to stick the sign to a ferrous surface.

Adafruit IO

We are using Adafruit IO to receive our text messages and send them to the Matrix Portal via WiFi.

Adafruit IO is a platform designed (by Adafruit!) to display, respond, and interact with your project's data. Data is kept private (data feeds are private by default) and secure (we will never sell or give this data away to another company) for you. It's the internet of things - for everyone!

What can Adafruit IO do for me?

Adafruit.io is a cloud service - that just means we run it for you and you don't have to manage it. You can connect to it over the Internet. It's meant primarily for storing and then retrieving data but it can do a lot more than just that!

- Display your data in real-time, online.
  - and share that data with others!
• Make your existing electronics project internet-connected: Control motors, read sensor data, and more!
• Connect projects to web services like Slack, Mastodon, Discord, RSS Feeds, Weather Services
• Connect your project to other internet-enabled devices
• Create No-Code electronics projects that connect to the internet.

This project needs an upgrade to Adafruit IO+, which costs $10/month or $99 per year, and allows you to send up to 25 SMS messages per day.

Get Started with Adafruit IO

Adafruit IO is integrated with your adafruit.com account so you don't need to create yet another online account! You need an Adafruit account to use Adafruit IO because we want to make sure the data you upload is available to only you (unless you decide to publish your data).

I have an Adafruit.com Account already

If you already have an Adafruit account, then you already have access to Adafruit IO. It doesn't matter how you signed up, your account will make all three available.

To access Adafruit IO, simply visit https://io.adafruit.com to start streaming, logging, and interacting with your data.

Create an Adafruit Account (for Adafruit IO)

An Adafruit account makes Adafruit content and services available to you in one place. Your account provides access to the Adafruit shop, the Adafruit Learning System, and Adafruit IO. This means only one account, one username, and one password are necessary to engage with the content and services that Adafruit offers.

If you do not have an Adafruit account, signing up for a new Adafruit account only takes a couple of steps.


Click the Sign Up button under the "Need An Adafruit Account?" title, below the Sign In section.
This will take you to the Sign Up page.

Fill in the requested information, and click the Create Account button.

**SIGN UP**

The best way to shop with Adafruit is to create an account which allows you to shop faster, track the status of your current orders, review your previous orders and take advantage of our other member benefits.

This takes you to your Adafruit Account home page. From here, you can access all the features of your account.
You can also access the Adafruit content and services right from this page. Along the top of the page, you'll see a series of links beginning with "Shop". To access any of these, simply click the link.

For example, to begin working with Adafruit IO, click the IO link to the right of the rest of the links. This is the same for the other links as well.

That's all there is to creating a new Adafruit account, and navigating to Adafruit IO.
Upgrade to Adafruit IO Plus

Your Adafruit IO experience is even better with IO+. The 'plus' stands for MORE STUFF! More feeds, dashboards, storage, speed. IO+ unlocks more data, more storage, and more feeds than our free service.

Upgrading your account to IO+ gives you:

- 60 data points per minute
- 60 days of data storage
- Actions every 5 seconds
- 25 SMS messages per day (UTC) for actions and the Messages Power-Up (USA and Canada)
- Unlimited dashboards
- Unlimited WipperSnapper devices
- Unlimited groups
- Unlimited feeds

Subscribe to Adafruit IO Plus

Ready to subscribe to Adafruit IO+? Log into your Adafruit account and visit io.adafruit.com/plus ( ).
Then, fill out your payment information, and your Adafruit IO account will be upgraded to IO Plus!

**IO Plus FAQ**

**Why should I subscribe to IO Plus? How is it billed?**

IO+, or IO Plus, is our enhanced version of IO. For one, uncomplicated price you get unlimited groups, feeds, WipperSnapper devices, and dashboards, a higher data input rate, and longer data storage.

IO+ subscriptions are billed immediately when you subscribe and then monthly or yearly, repeating each month or year on the same day you signed up until you cancel.

**Payments and Credit Card Processing**

We use Stripe for payment processing and subscription management and are currently able to accept every major credit card including Visa, Master Card, American Express, Discover, JCB, and Diners Club. All charges are denominated in USD (United States dollars) but international customers are welcome!

**What if I want to cancel my IO+ Subscription?**

It's easy and painless - when you cancel your Adafruit IO+ subscription, your access to IO+ will continue until the end of your current billing cycle. At then end of your subscription period you account will be restored to IO Free account limits. Right now that's 10 feeds, 5 dashboards, 30 days of data storage, and a 30 data points per minute data rate. Data that has already been stored will be preserved, and all extra feeds and dashboards will be archived.
I need a higher rate limit, is this possible?

If 60 data points per minute and 60 days of storage aren't enough, we offer additional account boosts. For only $2 per month, you can add 10 data points per minute and for $5 per month, you can add 30 extra days of storage. Add as many boosts as you need!

I'm having trouble with billing, or payment processing. Can you help?

Absolutely! If you have a question we haven't answered here or on the Adafruit IO forums, or if you have a question about your subscription specifically, you can use the Adafruit Contact page to get in touch with our support team directly.

Is there a way to purchase IO+ for a yearly fee instead of billing monthly?

Sure, we offer a one-year subscription pass in the Adafruit store.

Can I gift an IO Plus subscription to my friend, coworker, family-member, or a fellow maker?

You can purchase the subscription pass for yourself or as a gift, and send it to a family or friend. The code is redeemable within https://io.adafruit.com once you are signed in to your account. The code is not locked to any specific account until it is redeemed.

Enable SMS

Once you have Adafruit IO+ up and running, navigate to the Power-Ups tab and find the SMS option. Click "Get Started".
Click "Set up your Verified Phone Number" and follow the prompts to enter the number you want to use as your texting phone.

This will only work with one verified phone number per Adafruit IO+ account so be sure you're using the phone you want to use to text the sign.

Once the number is set up, click "Activate Phone Number" and follow the prompts to activate.

The activation window will give you a phone number to send your text messages to. Save this in your phone as a contact called "Adafruit IO" so you can find it later.

Create Feeds

Here is a guide describing how to set up feeds at Adafruit IO. However, since SMS is set up, there is an easier way to do this. Just set up feeds by texting the number received in the last step.
Send a text message to the number you were given in the last step with the name of the feed you want to set up. Keep it simple -- you'll need to type this name every time you update the sign. I called mine "text".

Adafruit IO will text back with the message "No data", telling us that we now have an empty feed called "text" with no data in it.

Send a second message in the format "feed name, data" -- in my case, I sent "Text, hello world". This creates an entry in the "Text" feed I just set up with a value of "hello world".

"Hello world" will now be the message that scrolls across the panel.
Let's see if that worked. Head to your Adafruit IO account and click the "feeds" tab. You should see a new feed called "Text" with a value of "Hello World".

Adding More Messages

To add more values to this feed, send more text messages in the format "feed, value" -- so,

- text, drink specials $10
- text, homework time
- text, home in 20 minutes

The most recent message you sent will appear on the matrix.

Color Choices

The code also gives options for color choices. Create called "color" and populate it with your favorite colors in hex format. Send the hex code (including the #), not the color name. I've included some common ones here for reference.

The code as written will choose a random color from your list each time the message scrolls.

- Red: #FF0000
- Green: #00FF00
- Blue: #0000FF
- Yellow: #FFFF00
- Cyan: #00FFFF
- Magenta: #FF00FF
- White: #FFFFFF
- Gray: #808080
- Orange: #FFA500
Matrix Portal Code

The very first run from the factory of the MatrixPortal S3 did not have the UF2 bootloader installed. Follow these instructions to install the UF2 bootloader: https://learn.adafruit.com/adafruit-matrixportal-s3/factory-reset#factory-reset-and-bootloader-repair-3107941

This project is based on the Matrix Portal Animated Message Board () code by Melissa LeBlanc Williams.

This code will support most of the features detailed in that guide. This guide shows how to add fonts and colors, but with a little creative coding you can also add background images, bitmaps, advanced scrolling and more.

To get the code up and running:

1. Reload the UF2 bootloader on the Matrix Portal S3 (if necessary, see above)
2. Install CircuitPython on the Matrix Portal S3
3. Connect to your local WiFi network
4. Upload the code and library bundle
5. Customize the code with your Matrix size and preferences
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 :)

This project requires version 9 or later of CircuitPython.

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 Downloads folder or desktop (or wherever is handy).
Plug your MatrixPortal S3 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.

Click the Reset button (indicated by the green arrow) on your board. When you see the NeoPixel RGB LED (indicated by the magenta arrow) turn purple, press it again. At that point, the NeoPixel should 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 MATRXS3BOOT.

Drag the adafruit_circuitpython_etc.uf2 file over to MATRXS3BOOT.

The LED will flash. Then, the MATRXS3BOOT drive will disappear and a new disk drive called CIRCUITPY will appear.

That's it, you're done! :)
Connect to WiFi

Next we'll get our WiFi and Adafruit IO information onto our Matrix Panel.

CircuitPython works with WiFi-capable boards to enable you to make projects that have network connectivity. This means working with various passwords and API keys. As of CircuitPython 8, there is support for a settings.toml file. This is a file that is stored on your CIRCUITPY drive, that contains all of your secret network information, such as your SSID, SSID password and any API keys for IoT services. It is designed to separate your sensitive information from your code.py file so you are able to share your code without sharing your credentials.

CircuitPython previously used a secrets.py file for this purpose. The settings.toml file is quite similar.

Your settings.toml file should be stored in the main directory of your CIRCUITPY drive. It should not be in a folder.

CircuitPython settings.toml File

This section will provide a couple of examples of what your settings.toml file should look like, specifically for CircuitPython WiFi projects in general.

The most minimal settings.toml file must contain your WiFi SSID and password, as that is the minimum required to connect to WiFi. Copy this example, paste it into your settings.toml, and update:

- `your_wifi_ssid`
- `your_wifi_password`

```
CIRCUITPY_WIFI_SSID = "your_wifi_ssid"
CIRCUITPY_WIFI_PASSWORD = "your_wifi_password"
```
Many CircuitPython network-connected projects on the Adafruit Learn System involve using Adafruit IO. For these projects, you must also include your Adafruit IO username and key. Copy the following example, paste it into your settings.toml file, and update:

- `your_wifi_ssid`
- `your_wifi_password`
- `your_aio_username`
- `your_aio_key`

```
CIRCUITPY_WIFI_SSID = "your_wifi_ssid"
CIRCUITPY_WIFI_PASSWORD = "your_wifi_password"
AIO_USERNAME = "your_aio_username"
AIO_KEY = "your_aio_key"
```

Some projects use different variable names for the entries in the settings.toml file. For example, a project might use `AIO_ID` in the place of `AIO_USERNAME`. If you run into connectivity issues, one of the first things to check is that the names in the settings.toml file match the names in the code.

Not every project uses the same variable name for each entry in the settings.toml file! Always verify it matches the code.

settings.toml File Tips

Here is an example settings.toml file.

```
# Comments are supported
CIRCUITPY_WIFI_SSID = "guest wifi"
```
In a settings.toml file, it's important to keep these factors in mind:

- Strings are wrapped in double quotes; ex: "your-string-here"
- Integers are not quoted and may be written in decimal with optional sign (+1, -1, 1000) or hexadecimal (0xabcd).
  - Floats, octal (0o567) and binary (0b11011) are not supported.
- Use \u escapes for weird characters, \x and \ooo escapes are not available in .toml files
  - Example: \U0001f44d for (thumbs up emoji) and \u20ac for € (EUR sign)
- Unicode emoji, and non-ASCII characters, stand for themselves as long as you're careful to save in "UTF-8 without BOM" format

When your settings.toml file is ready, you can save it in your text editor with the .toml extension.

Accessing Your settings.toml Information in code.py

In your code.py file, you'll need to import the os library to access the settings.toml file. Your settings are accessed with the os.getenv() function. You'll pass your settings entry to the function to import it into the code.py file.

```python
import os
print(os.getenv("test_variable"))
```
In the upcoming CircuitPython WiFi examples, you'll see how the settings.toml file is used for connecting to your SSID and accessing your API keys.

---

**Code the Message Board**

Once you've finished setting up your MatrixPortal S3 with CircuitPython, you can access the code, fonts, images and necessary libraries by downloading the Project Bundle.

To do this, click on the Download Project Bundle button in the window below. It will download as a zipped folder.

---

**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.

---

**Upload the Code, Fonts, Images and Libraries to the MatrixPortal S3**

```python
# SPDX-FileCopyrightText: 2023 Melissa-LeBlanc-Williams for Adafruit Industries
# SPDX-FileCopyrightText: 2020 John Park for Adafruit Industries
#
# SPDX-License-Identifier: MIT

# Quote board matrix display
# uses AdafruitIO to serve up a quote text feed and color feed
# random messages are displayed, updates periodically to look for new messages
# avoids repeating the same quote twice in a row

import time
import random
from collections import deque
import board
from adafruit_matrixportal.matrix import Matrix
from adafruit_matrixportal.network import Network
from messageboard import MessageBoard
from messageboard.fontpool import FontPool
from messageboard.message import Message
```
WIDTH = 64
HEIGHT = 32
DEFAULT_COLOR = 0x0000FF
DEFAULT_MESSAGE = "Text Adafruit IO to update"
DEFAULT_FONT = "arial_sm"
MESSAGES_FEED = "text"
COLORS_FEED = "color"
UPDATE_DELAY = 50   # Seconds between updates
SCROLL_DURATION = 5 # Seconds to scroll entire message
RANDOMIZE_FONTS = True # Randomize fonts, make "False" to just use the default font
RANDOMIZE_COLORS = True # Randomise colors, make "False" to just use the default color
KEEP_LATEST_MESSAGE = True # Keep the last message in the Message Feed

# --- Display setup ---
matrix = Matrix(width=WIDTH, height=HEIGHT, bit_depth=5)
network = Network(status_neopixel=board.NEOPixel, debug=True)
messageboard = MessageBoard(matrix)
fontpool = FontPool()
fontpool.add_font("arial_sm", "fonts/Arial-10.pcf")
fontpool.add_font("arial_lg", "fonts/Arial-15.pcf")
fontpool.add_font("sofia", "fonts/Sofia-Regular-15.pcf")

system_message = Message(fontpool.find_font("arial_sm"))
message_fonts = ("sofia", "arial_lg")
message_queue = deque((), 10000)  # Use a double-ended queue for messages
colors = []
def update_data():
    print("Updating data from Adafruit IO")
    # Only show connecting message if not connected
    if not network.is_connected:
        system_message.clear()
        system_message.add_text("Connecting", color=0xFFFF00)
        messageboard.animate(system_message, "Static", "show")
    try:
        color_data = network.get_io_data(COLORS_FEED)
        colors.clear()
        for json_data in color_data:
            color = network.json_traverse(json_data, ["value"])  
            colors.append(int(color[1:], 16))
    # pylint: disable=broad-except
    except Exception as error:
        print(error)
    try:
        messages_data = network.get_io_data(MESSAGES_FEED)
        message_ids = []
        messages = [] # Temporary place for messages
        for json_data in messages_data:
            message_ids.append(network.json_traverse(json_data, ["id"]))
            messages.append(network.json_traverse(json_data, ["value"]))
        # Results are returned in reverse order, so we reverse that and add to end
        of queue
        messages.reverse()
        for m in messages:
            message_queue.append(m)
            # Remove any messages that have been grabbed except the latest one if
            setting enabled
            start_index = 1 if KEEP_LATEST_MESSAGE else 0
            for index in range(start_index, len(message_ids)):
                message_id = message_ids[index]

©Adafruit Industries
```
# pylint: disable=broad-except
except Exception as error:
    print(error)

messageboard.animate(system_message, "Static", "hide")

def get_new_rand_item(current_index, item_list):
    if not item_list:
        return None
    if len(item_list) > 1 and current_index is not None:
        new_index = current_index
        while new_index == current_index:
            new_index = random.randrange(0, len(item_list))
    else:
        new_index = random.randrange(0, len(item_list))
    return new_index

update_data()
last_update = time.monotonic()
quote_index = None
color_index = None
font_index = None
message_text = None

while True:
    if len(message_queue) >= 1:
        message_text = message_queue.popleft()
    if message_text is None:
        message_text = DEFAULT_MESSAGE
    # Choose a random color from colors
    if RANDOMIZE_COLORS:
        color_index = get_new_rand_item(color_index, colors)
        if color_index is None:
            message_color = DEFAULT_COLOR
        else:
            message_color = colors[color_index]
    else:
        message_color = DEFAULT_COLOR
    # Choose a random font from message_fonts
    if RANDOMIZE_FONTS:
        font_index = get_new_rand_item(font_index, message_fonts)
        if font_index is None:
            message_font = DEFAULT_FONT
        else:
            message_font = message_fonts[font_index]
    else:
        message_font = DEFAULT_FONT
    # Set the quote text and color
    message = Message(fontpool.find_font(message_font))
    message.add_text(message_text, color=message_color)
    # Scroll the message
    duration = SCROLL_DURATION
    messageboard.animate(message, "Scroll", "right_to_left",
                          duration=SCROLL_DURATION)
    if time.monotonic() > last_update + UPDATE_DELAY:
        update_data()
        last_update = time.monotonic()
```
After downloading the Project Bundle, plug your MatrixPortal S3 into the computer's USB port. You should see a new flash drive appear in the computer's File Explorer or Finder (depending on your operating system) called CIRCUITPY. Unzip the folder and copy the following items to the MatrixPortal S3’s CIRCUITPY drive.

- lib folder
- fonts folder
- images folder
- code.py

Your MatrixPortal S3 CIRCUITPY drive should look like this after copying the fonts, images, and lib folders and the code.py and demo.py files.

Customize your Code

Find the variables near the top. You can set up the matrix size, and preferences for how you want the board to work.

```python
WIDTH = 64
HEIGHT = 32
```
DEFAULT_COLOR = 0x0000FF
DEFAULT_MESSAGE = "Text Adafruit IO to update"
DEFAULT_FONT = "arial_sm"

Width and height should match the number of total pixels in your Matrix Panel.

You can set your default colors and fonts here, as well as an intial message that plays the first time you turn on the panel.

Feed Settings

Next, set up the names of the feeds you set up on Adafruit IO - one for the messages, and one for your color repository.

MESSAGES_FEED = "text"
COLORS_FEED = "color"

Finally, set up the behavior of your panel. If KEEP_LATEST_MESSAGE is set to True, then the last thing scrolling on your panel when you power it down will be remembered. If it's set to False, then the default message will play on startup.

UPDATE_DELAY = 80   # Seconds between updates
SCROLL_DURATION = 10 # Seconds to scroll entire message
RANDOMIZE_FONTS = True  # Randomize fonts, make "False" to just use the default font
RANDOMIZE_COLORS = True # Randomise colors, make "False" to just use the default color
KEEP_LATEST_MESSAGE = True # Keep the last message in the Message Feed

Font Setup

The bundle includes a few fonts, but you're easily able to add more fonts or change them around.

To learn more about creating fonts for CircuitPython projects, check out this guide: Custom Fonts for CircuitPython Displays

Going Deeper

You can experiment with adding layers, images, and other scroll behaviors as well. There is a comprehensive guide about the myriad of features this code base offers.
MatrixPortal CircuitPython Animated Message Board
By M. LeBlanc-Williams

https://learn.adafruit.com/matrixportal-circuitpython-animated-message-board