Matrix Portal Scoreboard

Created by John Park

https://learn.adafruit.com/matrix-portal-scoreboard

Last updated on 2024-06-03 03:14:50 PM EDT

Table of Contents

Overview 3

- Parts

Install CircuitPython 5

- Set up CircuitPython Quick Start!
- Further Information

Prep the MatrixPortal 8

- Power Prep
- Power Terminals
- Panel Power
- Dual Matrix Setup
- Board Connection

LED Matrix Diffuser 12

- LED Diffusion Acrylic
- Measure and Cut the Plastic
- Uglu Dashes
- Stand

Code the Scoreboard 18

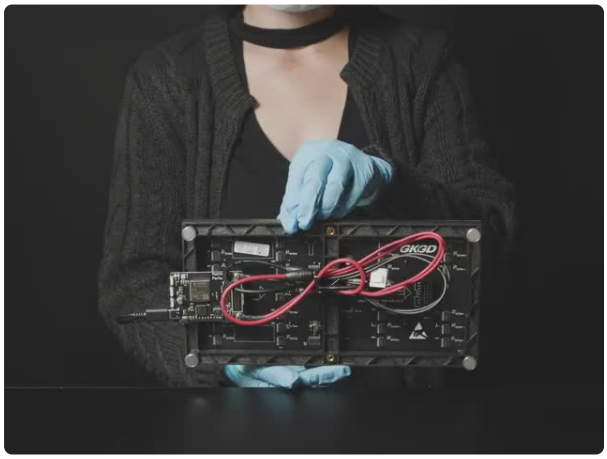
- Libraries
- Connect to the Internet
- Text Editor
- Code
- Adafruit IO Setup
- Problems Getting Score Data
- Adafruit IO Group, Feeds, Dashboard
- Group Creation
- Feed Creation
- Dashboard Creation
- Dashboard Widgets
- Score a Game
- How it Works
- Feed Variables
- Connecting Function
- Get Data Function
- Customize Team Names
- Update Scores Function
- Main Loop

Overview

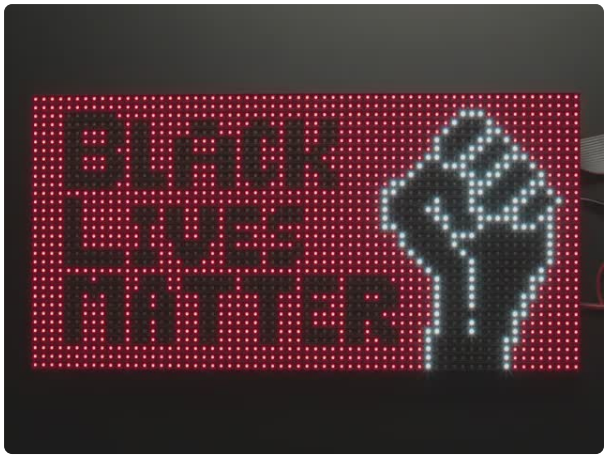
Are your backyard games of cornhole or horseshoes getting pretty serious? It may be time to build a professional-looking scoreboard!

Use the Matrix Portal and LED matrix display along with Adafruit IO to score a friendly competition between two players or teams! You can use your smart phone's web browser to adjust the scores in real-time. All coded in CircuitPython.

Parts



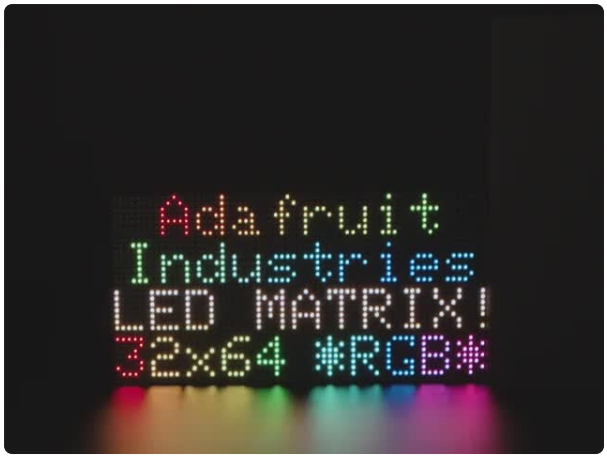
Adafruit Matrix Portal - CircuitPython Powered Internet Display
Folks love our wide selection of RGB matrices and accessories, for making custom colorful LED displays... and our RGB Matrix Shields...
https://www.adafruit.com/product/4745



64x32 RGB LED Matrix - 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



Black LED Diffusion Acrylic Panel 12" x 12" - 0.1" / 2.6mm thick

A nice whoppin' slab of some lovely black acrylic to add some extra diffusion to your LED Matrix project. This material is 2.6mm (0.1") thick and is made of special cast...

https://www.adafruit.com/product/4594

5V 2.5A Switching Power Supply with 20AWG MicroUSB Cable

Our all-in-one 5V 2.5 Amp + MicroUSB cable power adapter is the perfect choice for powering single-board computers like Raspberry Pi, BeagleBone, or anything else that's...

https://www.adafruit.com/product/1995

Micro B USB to USB C Adapter

As technology changes and adapts, so does Adafruit, and speaking of adapting, this adapter has a Micro B USB jack and a USB C...

https://www.adafruit.com/product/4299

USB cable - USB A to Micro-B
This here is your standard A to micro-B USB cable, for USB 1.1 or 2.0. Perfect for connecting a PC to your Metro, Feather, Raspberry Pi or other dev-board or...
https://www.adafruit.com/product/592

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!

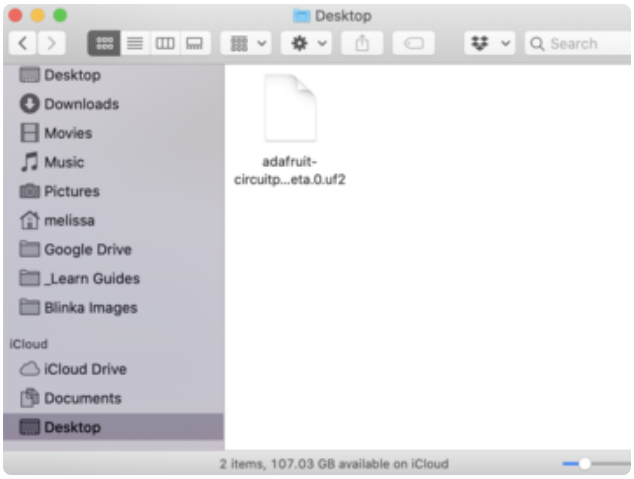
Follow this quick step-by-step for super-fast Python power :)

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

https://adafru.it/Nte

Further Information

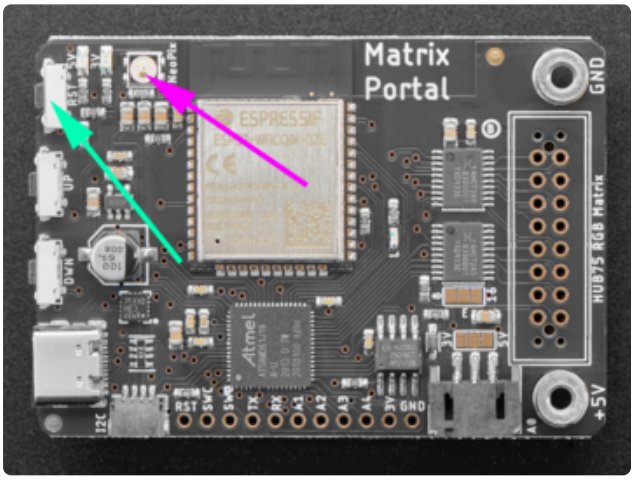
For more detailed info on installing CircuitPython, check out Installing CircuitPython (https://adafru.it/Amd).



Click the link above and download the latest UF2 file.

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

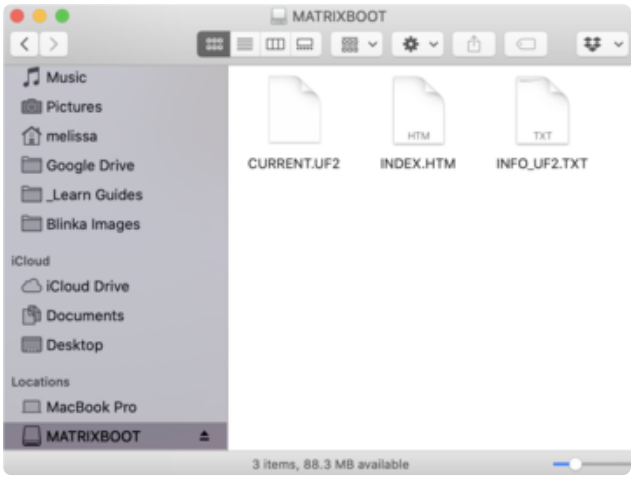
Plug your MatrixPortal M4 into your computer using a known-good USB cable.



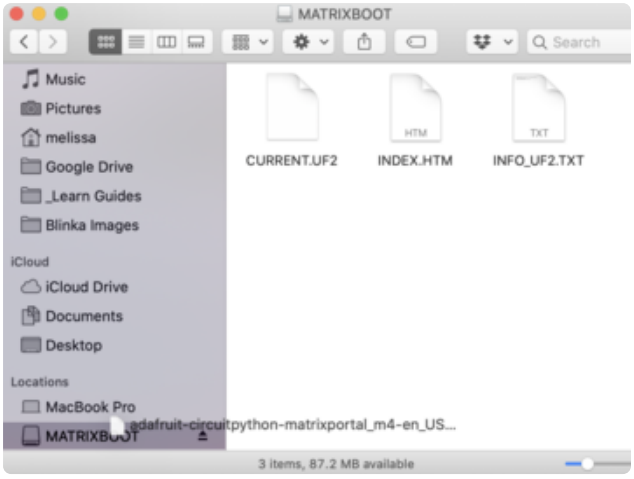
A lot of people end up using charge-only USB cables and it is very frustrating! So make sure you have a USB cable you know is good for data sync.

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.

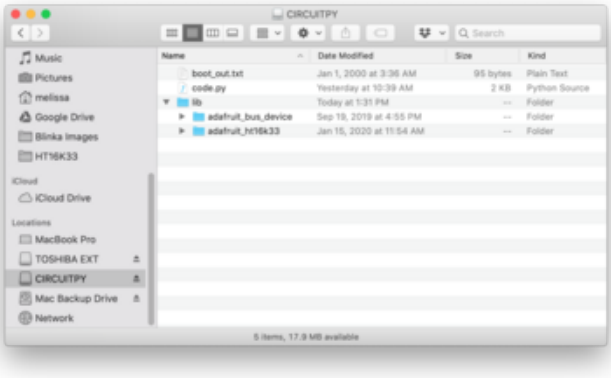
If double-clicking doesn't work the first time, try again. Sometimes it can take a few tries to get the rhythm right!



You will see a new disk drive appear called **MATRIXBOOT**.



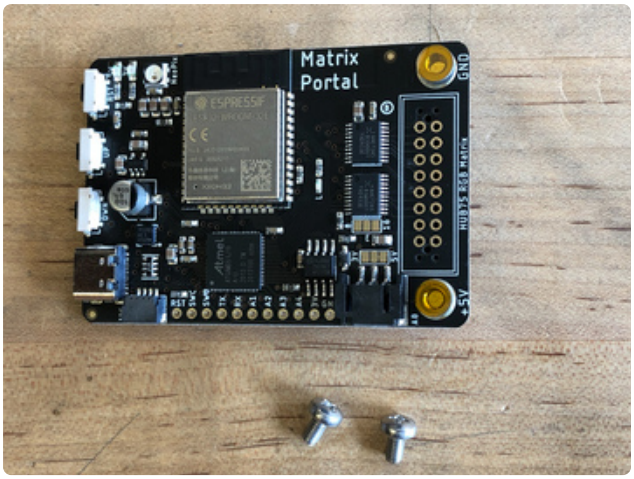
Drag the **adafruit_circuitpython_etc.uf2** file to **MATRIXBOOT.**

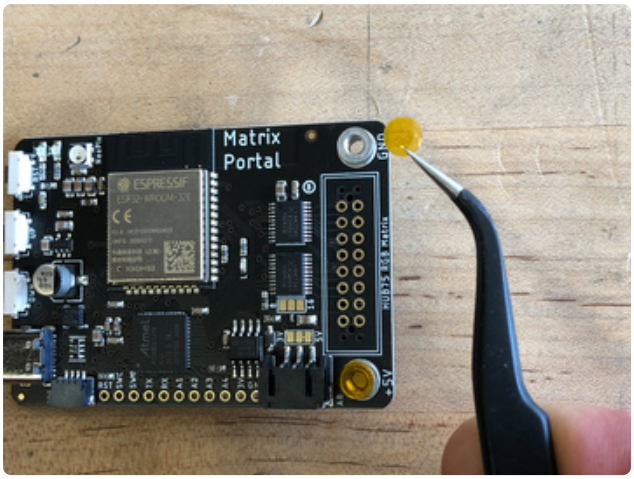


The LED will flash. Then, the **MATRIXBOOT** drive will disappear and a new disk drive called **CIRCUITPY** will appear.

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

Prep the MatrixPortal

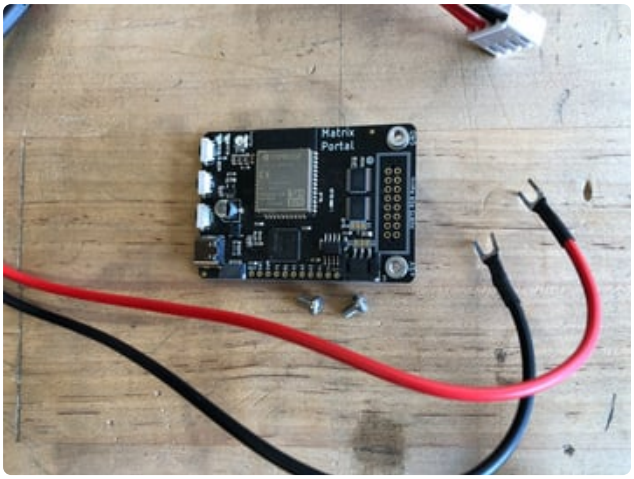


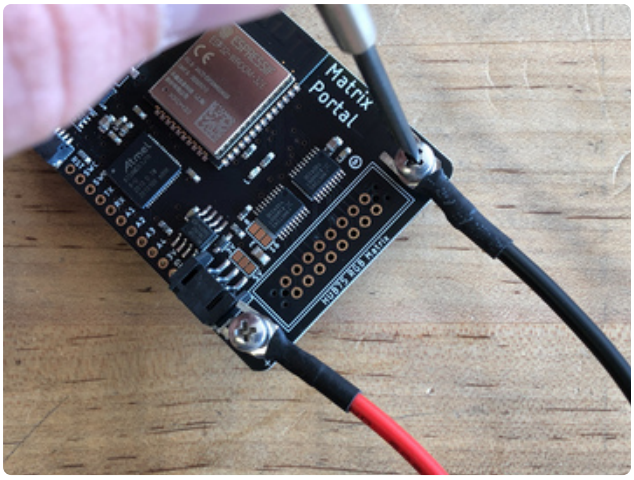


Power Prep

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.

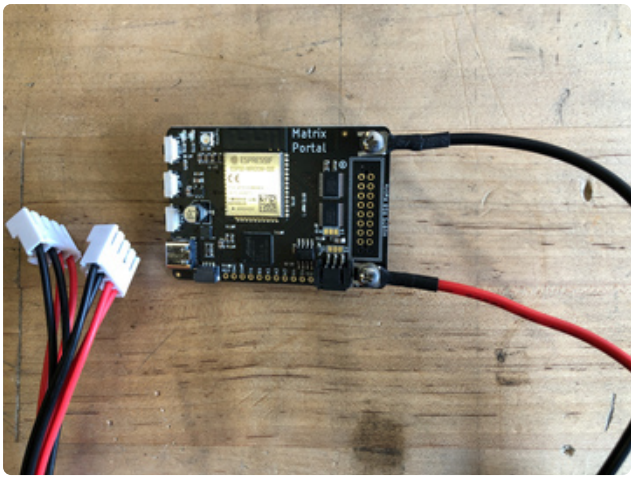


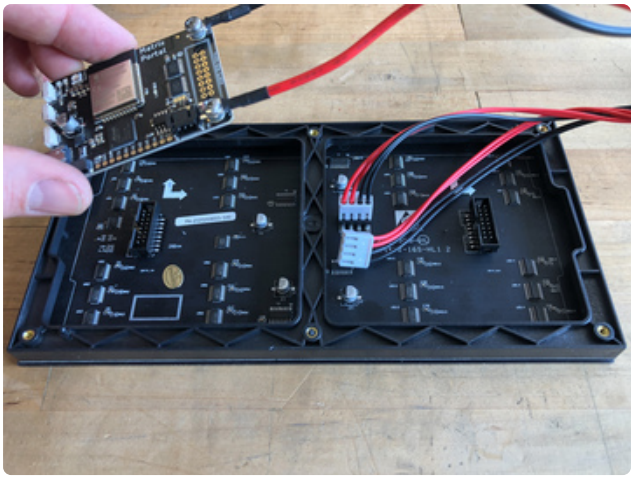


Power Terminals

Next, screw in the spade connectors to the corresponding standoff.

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



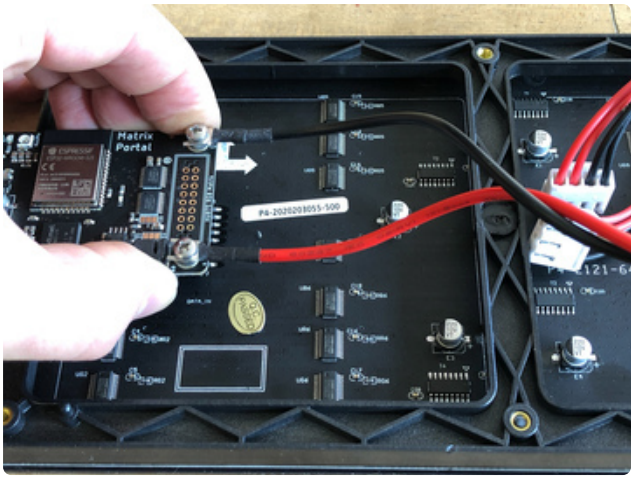


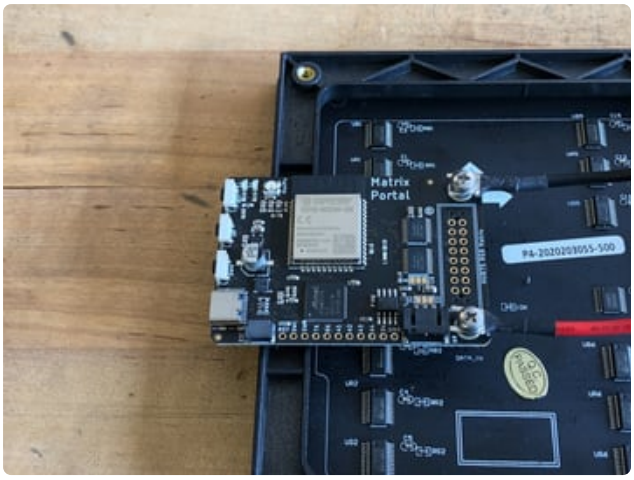
Panel Power

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 (https://adafru.it/OdJ).

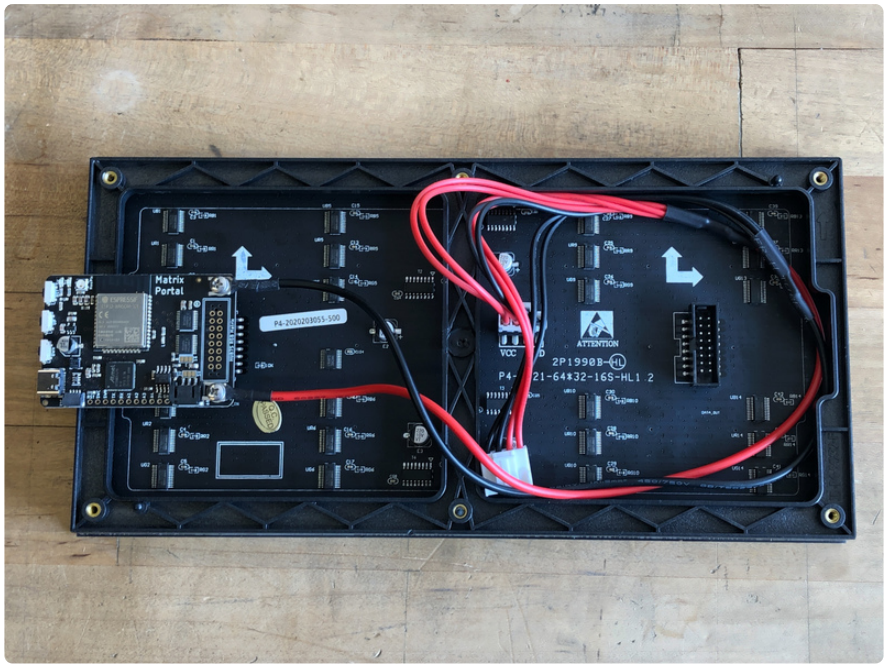




Board Connection

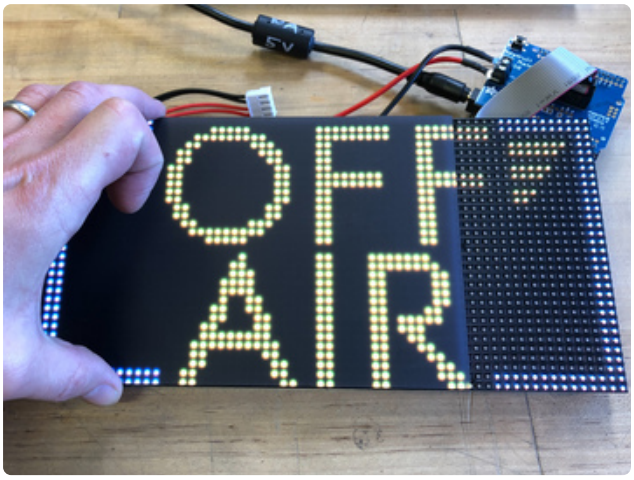
Now, plug the board into the left side shrouded 8x2 connector as shown. The orientation matters, so take a moment to confirm that the **white indicator arrow on the matrix panel is oriented pointing up and right** as seen here and the MatrixPortal overhangs the edge of the panel when connected. This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



For info on adding LED diffusion acrylic, see the page LED Matrix Diffuser.

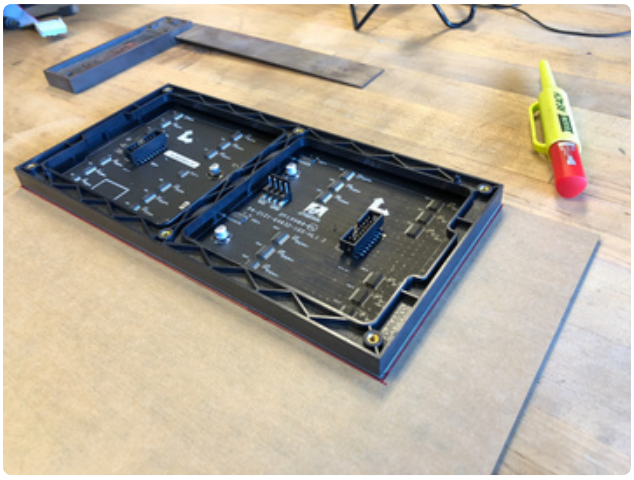
LED Matrix Diffuser

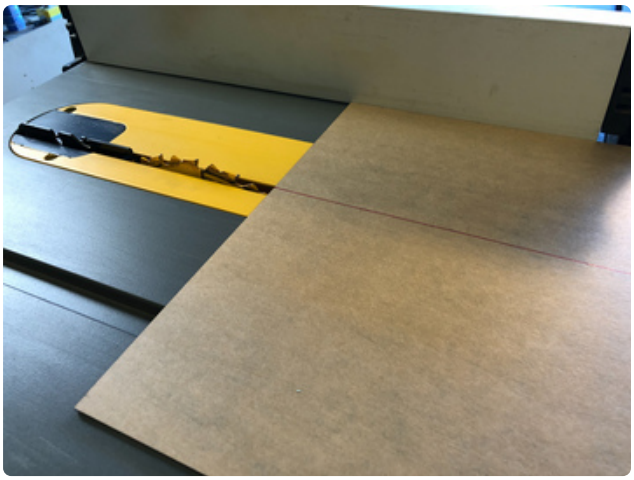


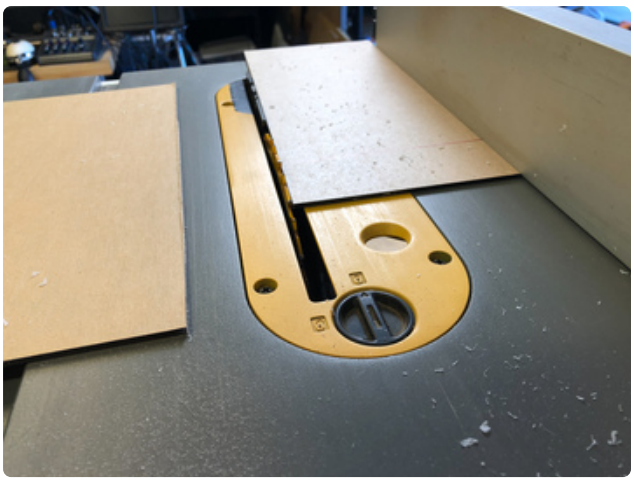
LED Diffusion Acrylic

You can add an LED diffusion acrylic faceplate (http://adafru.it/4594) to the your LED matrix display. (Pictured here with the ON AIR project (https://adafru.it/MPE))

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.





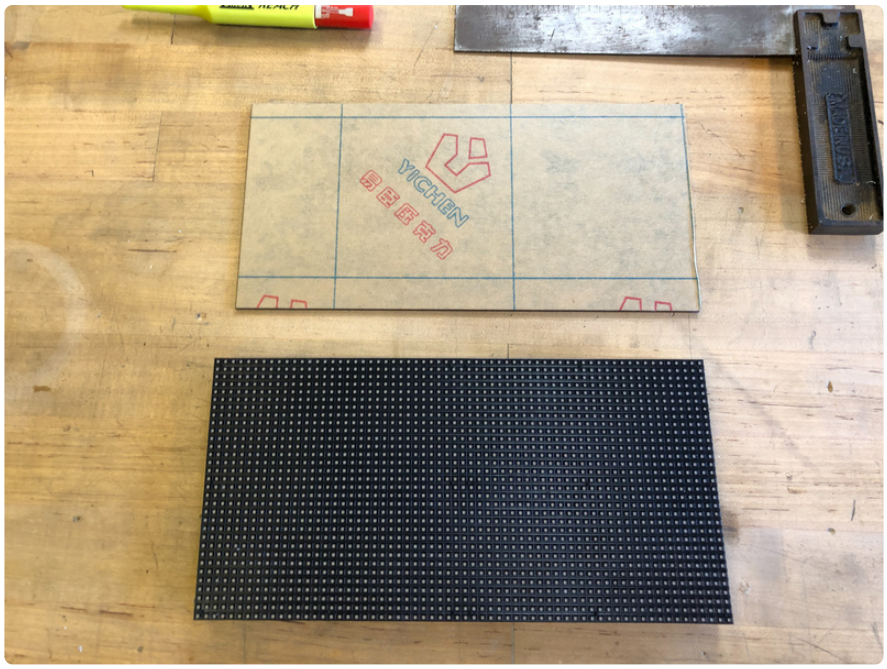


Measure and Cut the Plastic

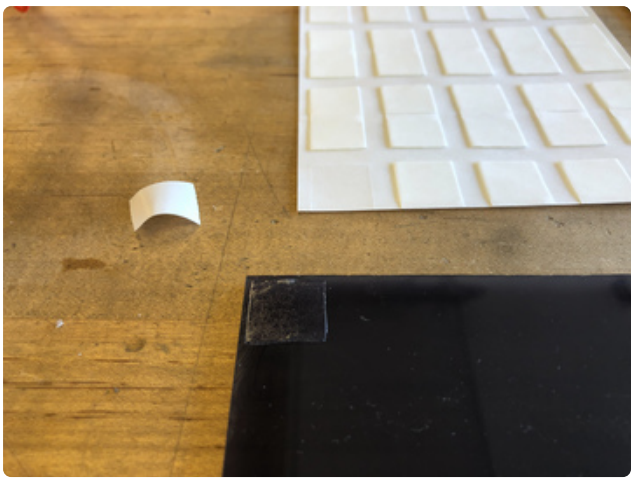
You can use the sign to measure and mark cut lines on the paper backing of the acrylic sheet.

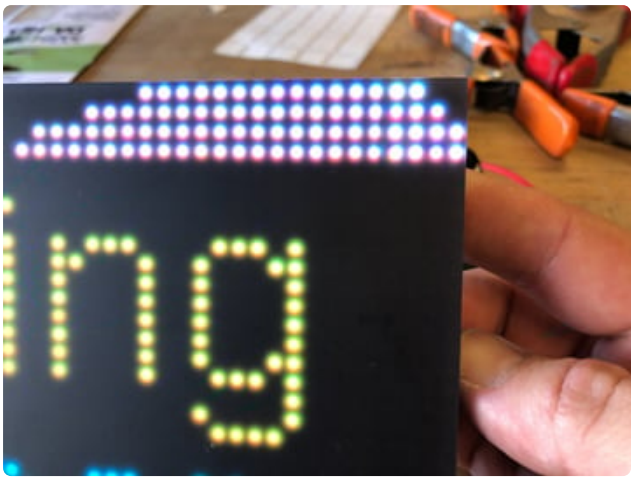
Then, use a tablesaw or bandsaw with a fine toothed blade and a guide or sled to make the cuts.

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 with the matte finished side facing out.



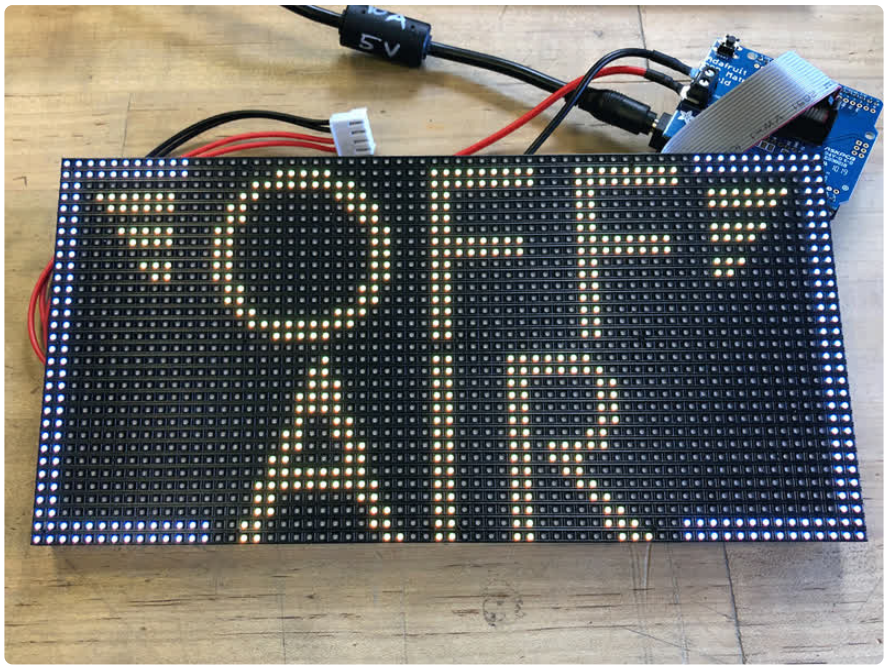


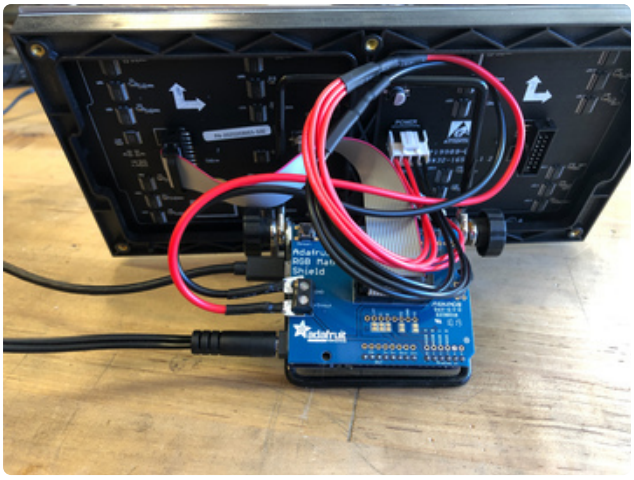
Uglu Dashes

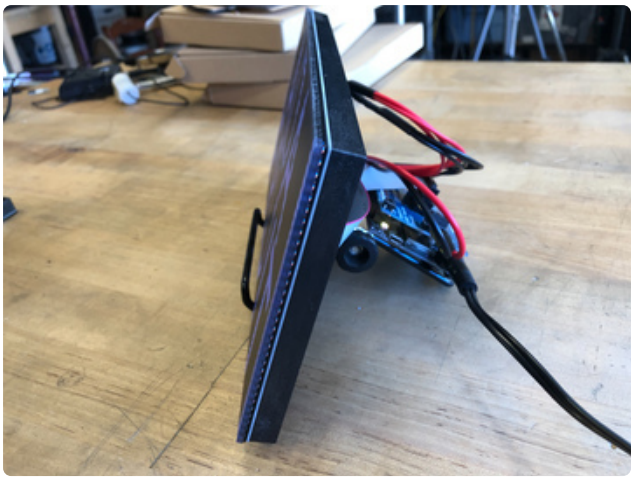
The best method we've found for adhering acrylic to the matrix display is to use Uglu Dashes clear adhesive rectangles from Pro Tapes (https://adafru.it/NcP). They are incredibly strong (although can be removed if necessary), easy to apply, and are invisible once attached.

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)


5V





Stand

A very simple and attractive way to display your matrix is with the adjustable bent-wire stand (http://adafru.it/1679).

Alternately, you can use a frame, 3D printed brackets (https://adafru.it/MZf), tape, glue, or even large binder clips to secure the acrylic to the sign and then mount it on on a wall, shelf, or display cabinet.

These mini-magnet feet (http://adafru.it/4631) can be used to stick the sign to a ferrous surface.

Code the Scoreboard

Libraries

We'll need to make sure we have these libraries installed. (Check out this link (https://adafru.it/ABU) on installing libraries if needed.)

adafruit_bitmap_font
adafruit_bus_device
adafruit_display_text
adafruit_esp32spi
adafruit_io
adafruit_matrixportal
adafruit_requests.mpy
neopixel.mpy

Connect to the Internet

Once you have CircuitPython setup and libraries installed we can get your board connected to the Internet. The process for connecting can be found here (https://adafru.it/NFK).

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).

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 it on the **CIRCUITPY** drive.

```
# SPDX-FileCopyrightText: 2020 John Park for Adafruit Industries
#
# SPDX-License-Identifier: MIT

# Scoreboard matrix display
# uses AdafruitIO to set scores and team names for a scoreboard
# Perfect for cornhole, ping pong, and other games

import time
import board
import terminalio
```

```
from adafruit_matrixportal.matrixportal import MatrixPortal

# --- Display setup ---
matrixportal = MatrixPortal(status_neopixel=board.NEOPIXEL, debug=False)

RED_COLOR = 0xAA0000
BLUE_COLOR = 0x0000AA

# Red Score
matrixportal.add_text(
    text_font=terminalio.FONT,
    text_position=(4, int(matrixportal.graphics.display.height * 0.75) - 3),
    text_color=RED_COLOR,
    text_scale=2,
)

# Blue Score
matrixportal.add_text(
    text_font=terminalio.FONT,
    text_position=(36, int(matrixportal.graphics.display.height * 0.75) - 3),
    text_color=BLUE_COLOR,
    text_scale=2,
)

# Red Team name
matrixportal.add_text(
    text_font=terminalio.FONT,
    text_position=(4, int(matrixportal.graphics.display.height * 0.25) - 4),
    text_color=RED_COLOR,
)

# Blue Team name
matrixportal.add_text(
    text_font=terminalio.FONT,
    text_position=(36, int(matrixportal.graphics.display.height * 0.25) - 4),
    text_color=BLUE_COLOR,
)

# Static 'Connecting' Text
matrixportal.add_text(
    text_font=terminalio.FONT,
    text_position=(59, 0),
)

SCORES_RED_FEED = "scores-group.red-team-score-feed"
SCORES_BLUE_FEED = "scores-group.blue-team-score-feed"
TEAM_RED_FEED = "scores-group.red-team-name"
TEAM_BLUE_FEED = "scores-group.blue-team-name"
UPDATE_DELAY = 4

matrixportal.set_text_color(RED_COLOR, 0)
matrixportal.set_text_color(BLUE_COLOR, 1)


def show_connecting(show):
    if show:
        matrixportal.set_text(".", 4)
    else:
        matrixportal.set_text(" ", 4)


def get_last_data(feed_key):
    feed = matrixportal.get_io_feed(feed_key, detailed=True)
    value = feed["details"]["data"]["last"]
    if value is not None:
        return value["value"]
    return None
```

```
def customize_team_names():
    team_red = "Red"
    team_blue = "Blue"

    show_connecting(True)
    team_name = get_last_data(TEAM_RED_FEED)
    if team_name is not None:
        print("Team {} is now Team {}".format(team_red, team_name))
        team_red = team_name
    matrixportal.set_text(team_red, 2)
    team_name = get_last_data(TEAM_BLUE_FEED)
    if team_name is not None:
        print("Team {} is now Team {}".format(team_blue, team_name))
        team_blue = team_name
    matrixportal.set_text(team_blue, 3)
    show_connecting(False)


def update_scores():
    print("Updating data from Adafruit IO")
    show_connecting(True)

    score_red = get_last_data(SCORES_RED_FEED)
    if score_red is None:
        score_red = 0
    matrixportal.set_text(score_red, 0)

    score_blue = get_last_data(SCORES_BLUE_FEED)
    if score_blue is None:
        score_blue = 0
    matrixportal.set_text(score_blue, 1)
    show_connecting(False)


customize_team_names()
update_scores()
last_update = time.monotonic()

while True:
    # Set the red score text
    if time.monotonic() > last_update + UPDATE_DELAY:
        update_scores()
        last_update = time.monotonic()
```

Adafruit IO Setup

Our project will use Adafruit IO to serve up a feed of quotes and colors. Adafruit IO is absolutely free to use, but you'll need to log in with your Adafruit account to use it. If you don't already have an Adafruit login, create one here (https://adafru.it/dAQ).

If you haven't used Adafruit IO before, check out this guide for more info (https://adafru.it/Ef8).

Once you have logged into your account, there are two pieces of information you'll need to place in your **secrets.py** file: **Adafruit IO username**, and **Adafruit IO key**. Head to io.adafruit.com (https://adafru.it/fsU) and simply click the **View AIO Key** link on the left hand side of the Adafruit IO page to get this information.

Then, add them to the **secrets.py** file like this:

```
secrets = {
    'ssid' : 'your_wifi_ssid',
    'password' : 'your_wifi_password',
    'aio_username' : '_your_aio_username_',
    'aio_key' : '_your_big_huge_super_long_aio_key_'
}
```

Problems Getting Score Data

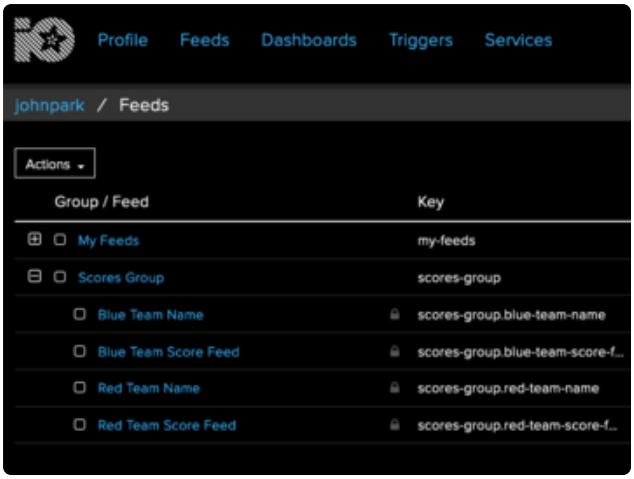
If you have any problems getting the data to update correctly, check that the **secrets.py** file has the information noted above.

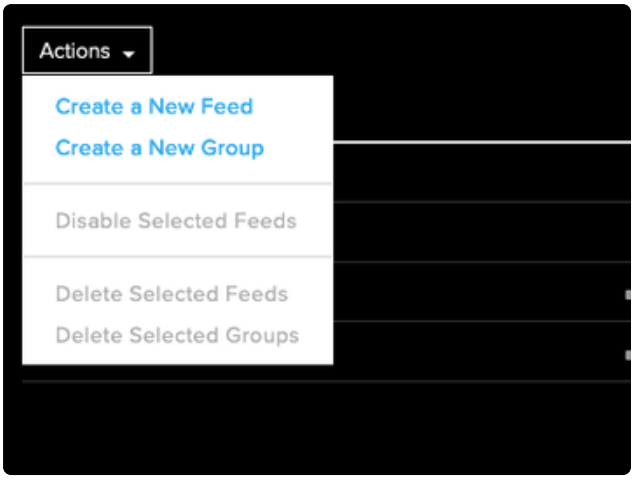
Adafruit IO Group, Feeds, Dashboard

Next, we'll create the necessary Adafruit IO Group, Feeds, and Dashboard to host our scores and team/player names.

First, if you're new to Adafruit IO, take a look at this excellent guide on getting started (https://adafru.it/BRB).

Next, we'll create a Group that will contain our four Feeds (two per player name, two per score).

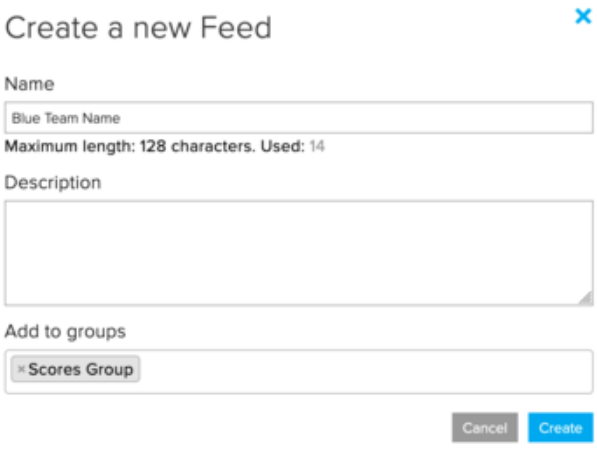




Group Creation

In the **Feeds** screen click on the **Actions** menu and then pick **Create a New Group** from the dropdown menu.

Name the group **Scores Group** and then click **Create**.



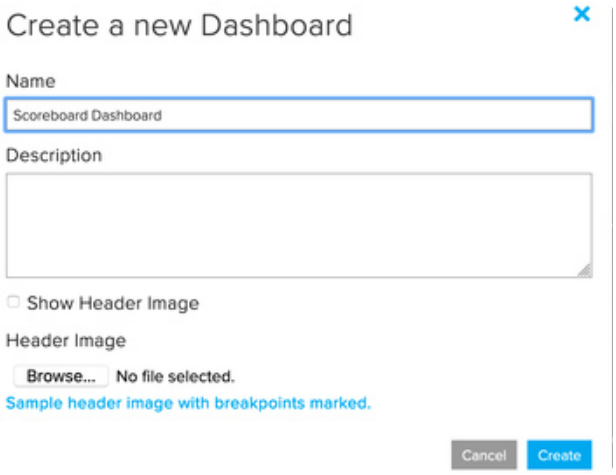
Feed Creation

In the **Feeds** screen click on the **Actions** menu and then pick **Create a New Feed** from the dropdown menu.

Name this feed **Blue Team Name,** and select **Scores Group** from the **Add to groups** field, then click **Create.**

Repeat this process a second time to make a new Feed called **Red Team Name.**

Repeat this process two more times to create feeds named **Blue Team Score Feed** and **Red Team Score Feed.**



Dashboard Creation

Now that we have the four feeds we need, let's create a Dashboard with a couple of UI block elements that will make it easy to add data points to our feeds.

From the Dashboards page, click the Actions dropdown menu and select **Create a New Dashboard**.

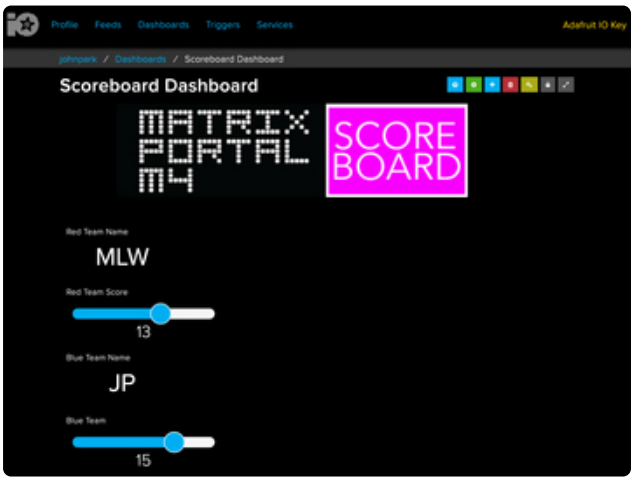
Name the Dashboard as **Scoreboard Dashboard.** You can make a custom head image to display by checking the checkbox for Show Header Image and browsing to an image file -- you can use the one shown below if you like.

Then click **Create**.

Dashboard Widgets

Click on the **create a new block** + sign in the Dashboard page, this will present you with a number of UI element block options.

Pick the **Text** block.

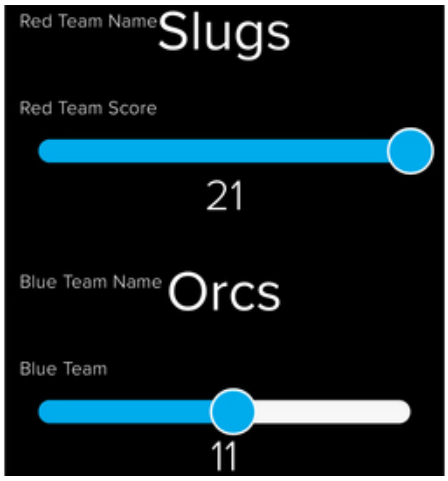


From the **Choose feed** pop-up window that appears, chose the **Blue Team Name** feed, then click **Next step**.

In the **Block settings** popup window, give the block the title **Text quote**, then click **Update block.**

Repeat this process to create a **Slider** block with a Min value of 0 and a Max value of 21 (or whatever your game calls for), assigning it to the **Blue Team Score Feed** feed.

Repeat the two steps from above for the **Red Team Name** and **Red Team Score Feed.**



Score a Game

Time to score a game! You will adjust those values from any web browser -- including a mobile device browser. This puts a centralized score keeping remote right in your hands while playing or officiating a game.

Start by setting the two team names, and set the scores to **0-0**.

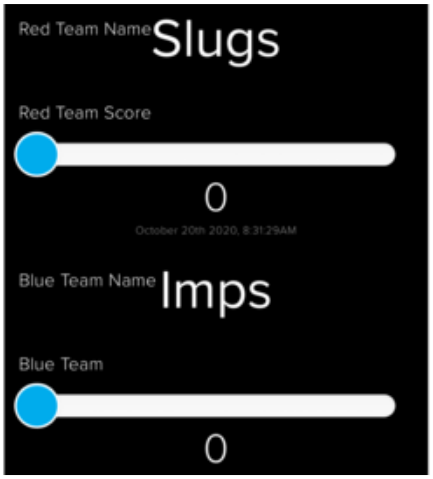
Now, any time a player or team scores, move the slider. After a few moments, the Matrix Scoreboard will update to match!

Here, the **Poughkeepsie Slugs** have beaten the **New Haven Orcs** with a final score of **21-11**!

Winner stays in the game, but a new contender emerges -- the **East Marion Imps** have made it all the way from Long Island for the game! In the **Dashboard**, we'll change the name from **Orcs** to Imps and press **Return** on the keyboard (or browser simulacrum thereof), then set the sliders to **0-0**.

You can press the reset button on the Matrix Portal so it will grab the new team names (these are only set at startup to keep things fast).

We're ready to start scoring the new game! Go Slugs!!



How it Works

Here's a look at the code and how it functions.

First, we load in some libraries:

```
import time
import board
import terminalio
from adafruit_matrixportal.matrixportal import MatrixPortal
```

Next, the Display is set up using the `matrixportal` object. We also define the team colors and create text label objects for the team names and scores and static connecting text:

```
matrixportal = MatrixPortal(status_neopixel=board.NEOPIXEL, debug=False)

RED_COLOR = 0xAA0000
BLUE_COLOR = 0x0000AA

# Red Score
matrixportal.add_text(
    text_font=terminalio.FONT,
    text_position=(4, int(matrixportal.graphics.display.height * 0.75) - 3),
    text_color=RED_COLOR,
    text_scale=2,
)

# Blue Score
matrixportal.add_text(
    text_font=terminalio.FONT,
    text_position=(36, int(matrixportal.graphics.display.height * 0.75) - 3),
    text_color=BLUE_COLOR,
    text_scale=2,
)

# Red Team name
matrixportal.add_text(
    text_font=terminalio.FONT,
    text_position=(4, int(matrixportal.graphics.display.height * 0.25) - 4),
    text_color=RED_COLOR,
)

# Blue Team name
matrixportal.add_text(
    text_font=terminalio.FONT,
    text_position=(36, int(matrixportal.graphics.display.height * 0.25) - 4),
    text_color=BLUE_COLOR,
)

# Static 'Connecting' Text
matrixportal.add_text(
    text_font=terminalio.FONT,
    text_position=(59, 0),
)
```

Feed Variables

In order to use the feed data, we need a few variables to help traverse the AIO json data, as well as set the update delay variable to 4 seconds.

We also set the text colors here.

```
SCORES_RED_FEED = "scores-group.red-team-score-feed"
SCORES_BLUE_FEED = "scores-group.blue-team-score-feed"
TEAM_RED_FEED = "scores-group.red-team-name"
TEAM_BLUE_FEED = "scores-group.blue-team-name"
UPDATE_DELAY = 4
```

```
matrixportal.set_text_color(RED_COLOR, 0)
matrixportal.set_text_color(BLUE_COLOR, 1)
```

Connecting Function

The `show_connecting()` function is used to display a dot symbol when the device is connecting to AIO over the WiFi access point.

```
def show_connecting(show):
    if show:
        matrixportal.set_text(".", 4)
    else:
        matrixportal.set_text(" ", 4)
```

Get Data Function

This function is used to pull in and parse the feed data from Adafruit IO.

```
def get_last_data(feed_key):
    feed = matrixportal.get_io_feed(feed_key, detailed=True)
    value = feed["details"]["data"]["last"]
    if value is not None:
        return value["value"]
    return None
```

Customize Team Names

This function pulls in the team names from the AIO feed.

```
def customize_team_names():
    team_red = "Red"
    team_blue = "Blue"

    show_connecting(True)
    team_name = get_last_data(TEAM_RED_FEED)
    if team_name is not None:
        print("Team {} is now Team {}".format(team_red, team_name))
        team_red = team_name
    matrixportal.set_text(team_red, 2)
    team_name = get_last_data(TEAM_BLUE_FEED)
    if team_name is not None:
        print("Team {} is now Team {}".format(team_blue, team_name))
        team_blue = team_name
    matrixportal.set_text(team_blue, 3)
    show_connecting(False)
```

Update Scores Function

This function is used to pull in the updated score data and then adjust the text to match.

```
def update_scores():
    print("Updating data from Adafruit IO")
    show_connecting(True)

    score_red = get_last_data(SCORES_RED_FEED)
    if score_red is None:
        score_red = 0
    matrixportal.set_text(score_red, 0)

    score_blue = get_last_data(SCORES_BLUE_FEED)
    if score_blue is None:
        score_blue = 0
    matrixportal.set_text(score_blue, 1)
    show_connecting(False)
```

The last steps of the setup are to run the `customize_team_names()` function and the `update_scores()` function, and then set a timekeeping variable.

```
customize_team_names()
update_scores()
last_update = time.monotonic()
```

Main Loop

The main loop of the program simply waits for the four second update delay to pass and then calls `update_scores()`, and resets the `last_update` time.

```
while True:
    # Set the red score text
    if time.monotonic() &gt; last_update + UPDATE_DELAY:
        update_scores()
        last_update = time.monotonic()
```