Stream Deck controlled RGB Message Panel using Adafruit IO

Created by Melissa LeBlanc-Williams


Last updated on 2022-01-26 02:43:32 PM EST
# 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>4</td>
</tr>
<tr>
<td>Stream Deck Plugin</td>
<td>6</td>
</tr>
<tr>
<td>• Installing the Plugin</td>
<td>6</td>
</tr>
<tr>
<td>• Configuring the Plugin</td>
<td>7</td>
</tr>
<tr>
<td>• Formatting Messages</td>
<td>8</td>
</tr>
<tr>
<td>• How it Works</td>
<td>8</td>
</tr>
<tr>
<td>Adafruit IO REST API</td>
<td>9</td>
</tr>
<tr>
<td>Arduino Code</td>
<td>9</td>
</tr>
<tr>
<td>• Install Arduino Libraries</td>
<td>9</td>
</tr>
<tr>
<td>• Download the Sketch</td>
<td>11</td>
</tr>
<tr>
<td>• Configuring the Sketch</td>
<td>15</td>
</tr>
<tr>
<td>3D Printing</td>
<td>15</td>
</tr>
<tr>
<td>• Single Piece for Newer Matrix</td>
<td>16</td>
</tr>
<tr>
<td>• Single Piece for Older Matrix</td>
<td>16</td>
</tr>
<tr>
<td>• Two Pieces for the Newer Matrix</td>
<td>17</td>
</tr>
<tr>
<td>• Two Pieces for the Older Matrix</td>
<td>18</td>
</tr>
<tr>
<td>• 3D Printer Settings</td>
<td>20</td>
</tr>
<tr>
<td>Assembly</td>
<td>20</td>
</tr>
<tr>
<td>Troubleshooting</td>
<td>25</td>
</tr>
</tbody>
</table>

©Adafruit Industries
Overview

If you've ever done any live video streaming before, you may have heard of the [Stream Deck](https://adafruit.it/FC0). It allows you to customize the live streaming experience for your viewers such as showing the number of viewers, running custom keyboard commands, or showing animations. It allows you to customize the buttons too with whatever graphics you want.

But did you know there's other non-streaming uses for the Stream Deck such as controlling lights in a house, open applications, or open web pages in a browser window? You can even write custom plugins to extend the capabilities.

This project uses a custom Stream Deck plugin to communicate directly with the Adafruit IO REST API. It works by posting a specific value to a feed and each button will post a different value to the feed allowing you to easily change messages.

The message panel itself is driven by the Metro M4 Express Airlift Lite. It will connect to local WiFi and read the value of the message straight from your feed. It even allows some basic formatting such as color and text size. The feed values are limited to 50 characters, so the more formatting you apply, the less text you can show. Fortunately, the message panel can hold a maximum of 40 characters.
Parts

Adafruit Metro M4 Express AirLift (WiFi) - Lite
Give your next project a lift with AirLift - our witty name for the ESP32 coprocessor that graces this Metro M4. You already know about the Adafruit Metro...
https://www.adafruit.com/product/4000

On top of the Metro M4 Express Airlift is an Adafruit RGB Matrix Shield (https://adafruit.com/FC1).

Adafruit RGB Matrix Shield for Arduino
Our RGB matrices are dazzling, with their hundreds or even thousands of individual RGB LEDs. Compared to NeoPixels, they’ve got great density, power usage and the...
https://www.adafruit.com/product/2601

For this project, the 64x32 RGB Matrix was used and it is available in 4 different pitches. The smaller the pitch is, the smaller the LEDs are, but also the closer together they are.

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
You will also need a 5V power adapter that can drive a lot of current. We stock a nice 4 Amp model which was used for this project.

[Image of 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

If you plan on 3D printing the Backing, you will also need these Heat-set Inserts.

[Image of Brass Heat-Set Inserts for Plastic - M3 x 3mm - 50 pack]

Wanna improve the connection strength between your project's 3D-printed parts, and also have nice clean surfaces? Instead of gluing bits together, or screwing plastic screws...

https://www.adafruit.com/product/4256

You also may want this tool for the heat-set inserts.

[Image of Heat-Set Insert For Soldering Irons - #4-40 / M3 Inserts]

Wanna improve the connection strength between your project's 3D-printed parts, and also have nice clean surfaces? Instead of gluing bits together, or screwing plastic screws...

https://www.adafruit.com/product/4239

4 x M3x6mm Screw

McMaster-Carr M3x6mm Socket head screw for mounting

Metro M4 Airlift

https://www.mcmaster.com/95263a110
M3x8mm should also work fine if that's what you have available.

The only other hardware you will need are some nice long wood screws (about 1.5" or longer) for mounting in the wall. It is very important that these are steel screws because the magnetic screws will need to stick to them.

Stream Deck Plugin

Stream Deck allows custom plugins to be written in several different languages. However, writing the plugin in JavaScript has several advantages for communicating with Adafruit IO. First JavaScript is a platform independent language, so the plugin should work on both Windows and MacOS systems. Second, because we are communicating with a website, JavaScript is a natural choice and makes this very easy. And finally, there are plenty of examples available in JavaScript.

In the initial prototyping phase of creating the Message Panel, I tried using an existing plugin that allows communicating with IFTTT webhooks, which can perform an action on Adafruit IO, but there were several problems with this. It was pretty slow and would sometimes take a few seconds to update, messages were difficult to change and required drilling down into the IFTTT interface, and also the IFTTT was stripping certain characters away.

The Stream Deck plugin communicates directly through the Adafruit IO REST API and overcomes all of those limitations. REST stands for REpresentational State Transfer and works by making GET, POST, PUT, PATCH, and DELETE calls, although PATCH isn’t as common as the other four. Each of these calls has a different purpose. For placing a value into an Adafruit IO feed, we only need the POST call. To learn more about REST, be sure to check out our All the Internet of Things - Episode Two: Protocols guide.

If you would like an excellent overview of Adafruit IO, be sure to check out our Welcome to Adafruit IO guide.

Installing the Plugin

First, make sure you are running the latest Stream Deck application for your computer. Next you can download it directly from this guide.

Download Adafruit IO Stream Deck Plugin
Once you have downloaded it, open it up and when it asks you if you would like to install it, click yes. Then either open up the Stream Deck application or if it is already running, choose the Configure Stream Deck option from your menu bar. The new plugin will appear on the right-hand side under Adafruit IO and be called Publish to Feed.

Configuring the Plugin

Go ahead and drag Publish to Feed over to the desired spot on your Stream Deck and it will be automatically selected. After that, there are four required pieces of information.

First you will need your Adafruit IO Username and Active Key. You can find these by logging into Adafruit IO at https://io.adafruit.com (https://adafruit.it/eZ8) and selecting the AIO Key link on the right side of the screen.

The other two pieces of information we need are the Feed Key and the Value that we want to publish. The feed key will need to be the Key of an existing feed. For more information on creating a feed, you can view our Adafruit IO Basics: Feeds (https://adafruit.it/ioA) guide page called Creating a Feed (https://adafruit.it/f5k). Once that is created, you can look at your list of feeds by selecting Feeds → View All. Type in the Feed Key, which is the value in the column labeled Key.
Formatting Messages

By default, the color is white and if you would like to format a message, there are two different options available. Text will always be automatically centered both horizontally and vertically.

Color

First you can change the color by supplying either a value between 0-23 that corresponds to a rainbow color on the color wheel or a number above 23 for white. The number should be placed inside of a matched set of braces such as \{4\} to select color 4 on the color wheel. You will need to experiment to see which colors look best for your application. It will continue printing in that color until you change it again.

Text Size

You can also set the text size in a similar way, but it should placed inside of a matched set of angled brackets such as \<2\> for size 2. Just like with color, it will continue printing in that color until you change it again.

How it Works

Besides the usual store/retrieve values code that is part of all Stream Deck plugins, it works by simply making an XMLHttpRequest post call to the Adafruit IO REST API. The IO Key is included in one of the header parameters, the username and feed key are part of URL, and the message is the main piece of data. Here's the relevant JavaScript code from the plugin:
const request = new XMLHttpRequest();
var datum = {
    "value": feedvalue
};

request.open("POST", 'https://io.adafruit.com/api/v2/' + username + '/feeds/' + feedkey + '/data');
request.setRequestHeader("X-AIO-Key", iokey);
request.setRequestHeader("Content-Type", "application/json");
request.send(JSON.stringify(datum));

You can read more about the Adafruit IO REST API in our handy reference guide (http://adafruit.io/uff).

Adafruit IO REST API
Adafruit IO REST API (https://adafruit.io/uff)

Arduino Code

The Arduino code is responsible for connecting to your local WiFi, connecting to the Adafruit IO service, retrieving the value stored in the feed, decoding the formatting and displaying the message on the RGB Panel. However, in order to use it, you will need to supply several parameters in the configuration.

Install Arduino Libraries

In order for Metro M4 Express Airlift to work, you will need to install several Arduino libraries. First, to use a variant of the Arduino WiFiNINA library, which is amazing and written by the Arduino team! Adafruit made a fork that you should install to get the functionality for the AirLift.

Click here to download the library:

Download Adafruit's version of WiFiNINA
https://adafruit.io/Evm

Within the Arduino IDE, select Install library from ZIP...
And select the zip you just downloaded.

Six more libraries need to be installed using the Arduino Library Manager...this is the preferred and modern way. From the Arduino “Sketch” menu, select “Include Library” then “Manage Libraries...”

Type “gfx” in the search field to quickly find the first library — Adafruit_GFX:

Repeat the search and install steps, looking for the Adafruit BusIO, Adafruit MQTT Library, RGB matrix Panel, ArduinoHttpClient, and Adafruit IO Arduino libraries.
Download the Sketch

Next you will need to download the sketch. Click on the following button to be taken to Github and download the .ino file and open it up in the Arduino IDE.

[Download Sketch]

The main sketch starts off by initializing the RGB and then connecting to Adafruit IO and listening for messages. When new messages come in, it calculates the starting point of each line with formatting applied first and then using those calculations, it draws and formats the message to the RGB Matrix Panel.

```c
// Message Panel
// Reads an Adafruit IO Feed, then formats and displays the message
// Author: Melissa LeBlanc-Williams

#include <RGBmatrixPanel.h>
#include <SPI.h>

#define BASE_CHAR_WIDTH 6  // pixels + 1 space
#define BASE_CHAR_HEIGHT 8 // pixels + 1 space

// Most of the signal pins are configurable, but the CLK pin has some special constraints. On 8-bit AVR boards it must be on PORTB...
// Pin 8 works on the Arduino Uno & compatibles (e.g. Adafruit Metro),
// Pin 11 works on the Arduino Mega. On 32-bit SAMD boards it must be on the same PORT as the RGB data pins (D2-D7)...
// Pin 8 works on the Adafruit Metro M0 or Arduino Zero,
// Pin A4 works on the Adafruit Metro M4 (if using the Adafruit RGB Matrix Shield, cut trace between CLK pads and run a wire to A4).

//#define CLK 8 // USE THIS ON ADAFRUIT METRO M0 or adapting to use Airlift, etc.
#define CLK A4 // USE THIS ON METRO M4 (not M0)
#define OE 9
#define LAT 10
#define A 0
#define B 1
#define C 2
#define D 3

RGBmatrixPanel matrix(A, B, C, D, CLK, LAT, OE, false, 64);

#include "config.h"

// set up the 'messagepanel' feed
AdafruitIO_Feed *counter = io.feed("messagepanel");

void drawText(const char *text, bool resetPosition = true, uint16_t color = 0xffff, uint16_t textSize = 1) {
  matrix.setTextSize(textSize); // size 1 == 8 pixels high
  if (resetPosition) {
    matrix.setCursor(0, 0); // start at top left, with 8 pixel of spacing
  }
  matrix.setTextColor(color);
  matrix.print(text);
}
```

©Adafruit Industries
void handleMessage(AdafruitIO_Data *data) {
// This function is called whenever a 'messagepanel' message
// is received from Adafruit IO. It was attached to
// the counter feed in the setup() function above.
String message = data->toString();
String plainText = data->toString();
uint16_t colorSize = matrix.Color333(7, 7, 7);
uint16_t textSize = 1;
uint16_t colorStartIndex = 0, colorEndIndex = 0;
uint16_t sizeStartIndex = 0, sizeEndIndex = 0;
uint16_t(strpos = 0);
byte lineLengths[] = {0, 0, 0, 0};
byte lineNumber = 0;
byte messageHeight = 0;
byte lineHeight = 0;
// Calculate line lengths
boolean paramRead = false;
boolean newLine = false;

matrix.setCursor(0, 0);
matrix.fillScreen(matrix.Color333(0, 0, 0));

// Strip out all color data first
while (strpos < plainText.length()) {
    colorStartIndex = plainText.indexOf('{');
    colorEndIndex = plainText.indexOf('}');
    plainText.remove(colorStartIndex, colorEndIndex - colorStartIndex + 1);
    strpos++;
}
// Calculate the line lengths in pixels for fixed width text
strpos = 0;
while (strpos < plainText.length()) {
    sizeStartIndex = plainText.indexOf('<');
    sizeEndIndex = plainText.indexOf('>');

    if (strpos == sizeStartIndex) {
        textSize = atoi(plainText.substring(sizeStartIndex + 1, sizeEndIndex).c_str());
        plainText.remove(sizeStartIndex, sizeEndIndex - sizeStartIndex + 1);
    }
    if (plainText.charAt(strpos) != '\n') {
        lineLengths[lineNumber] += textSize * BASE_CHAR_WIDTH;
        if (textSize * BASE_CHAR_HEIGHT > lineHeight) {
            lineHeight = textSize * BASE_CHAR_HEIGHT;
        }
    }
}
// We want to keep adding up the characters * textSize until we hit a newline character
// or we reach the width of the message panel. Then we go down to the next line
if (plainText.charAt(strpos) == '\n' || lineLengths[lineNumber] >= matrix.width()) {
    messageHeight += lineHeight;
    lineHeight = 0;
    lineNumber++;
    strpos++;
}
// Add the last line
messageHeight += lineHeight;
textSize = 1;
lineNum = 0;
for (uint16_t i = 0; i < message.length(); i++) {
if (message.charAt(i) == '{') {
    paramRead = true;
    colorStartIndex = i + 1;
} else if (message.charAt(i) == '}') {
    paramRead = false;
    int wheelPos = atoi(message.substring(colorStartIndex, i).c_str());
    if (wheelPos < 24) {
        color = Wheel(wheelPos);
    } else {
        color = matrix.Color333(7, 7, 7);
    }
} else if (message.charAt(i) == '<') {
    paramRead = true;
    sizeStartIndex = i + 1;
} else if (message.charAt(i) == '>') {
    paramRead = false;
    textSize = atoi(message.substring(sizeStartIndex, i).c_str());
} else {
    if (paramRead) continue;
    if (matrix.getCursorX() == 0 && matrix.getCursorY() == 0) {
        matrix.setCursor(floor((matrix.width() / 2) - (lineLengths[lineNum] / 2)),
                        matrix.height() / 2 - messageHeight / 2);
    } else if (newLine) {
        matrix.setCursor(floor((matrix.width() / 2) - (lineLengths[++lineNum] / 2)),
                        matrix.getCursorY());
        newLine = false;
    }
    drawText(message.substring(i, i + 1).c_str(), false, color, textSize);
    if (message.charAt(i) == '\n' || matrix.getCursorX() >= matrix.width()) {
        newLine = true;
    }
}
}

void setup() {
    matrix.begin();

    // fill the screen with 'black'
    matrix.fillScreen(matrix.Color333(0, 0, 0));
    // draw some text!
    matrix.setTextWrap(true);
    drawText("Connecting...");
    Serial.begin(115200);
    io.connect();

    counter->onMessage(handleMessage);

    while(io.mqttStatus() < AIO_CONNECTED) {
        drawText(".");
        delay(500);
    }
    counter->get();

    Serial.println();
    Serial.println(io.statusText());
}

void loop() {
    io.run();
}

// Input a value 0 to 23 to get a color value.
// The colours are a transition r - g - b - back to r.
uint16_t Wheel(byte WheelPos) {
    if(WheelPos < 8) {
        return matrix.Color333(7 - WheelPos, WheelPos, 0);
else if(WheelPos < 16) {
    WheelPos -= 8;
    return matrix.Color333(0, 7 - WheelPos, WheelPos);
} else {
    WheelPos -= 16;
    return matrix.Color333(WheelPos, 0, 7 - WheelPos);
}

One other file is used and that is config.h. This holds the connection information for WiFi and connecting to Adafruit IO. This is the same file is included with many of the Adafruit IO Library examples.

/************************************************ Adafruit IO Config ************************************************/

// visit io.adafruit.com if you need to create an account,
// or if you need your Adafruit IO key.
#define IO_USERNAME "your_username"
#define IO_KEY "your_key"

/************************************************ WIFI ************************************************/

// the AdafruitIO_WiFi client will work with the following boards:
//   - Feather M0 WiFi -> https://www.adafruit.com/products/3010
//   - Feather WICED -> https://www.adafruit.com/products/3056
//   - Adafruit Metro M4 Express AirLift Lite -> https://www.adafruit.com/product/4000

#define WIFI_SSID "your_ssid"
#define WIFI_PASS "your_pass"

// uncomment the following line if you are using airlift
// #define USE_AIRLIFT

// uncomment the following line if you are using winc1500
// #define USE_WINC1500

// comment out the following lines if you are using fona or ethernet
#define "AdafruitIO_WiFi.h"

#if defined(USE_AIRLIFT) || defined(ADAFRUIT_METRO_M4_AIRLIFT_LITE)
    // Configure the pins used for the ESP32 connection
    #if !defined(SPIWIFI_SS) // if the wifi definition isn't in the board variant
        // Don't change the names of these #define's! they match the variant ones
        #define SPIWIFI SPI
        #define SPIWIFI_SS 10 // Chip select pin
        #define NINA_ACK 9 // a.k.a BUSY or READY pin
        #define NINA_RESETN 6 // Reset pin
        #define NINA_GPIO0 -1 // Not connected
    #endif
    AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS, SPIWIFI_SS, NINA_ACK, NINA_RESETN, NINA_GPIO0, &SPIWIFI);
#else
    AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS);
#endif

/*************************************************************** FONA **************************************************************/

// the AdafruitIO_FONA client will work with the following boards:
Configuring the Sketch

There are a few things you will need to edit to get the message panel up and running. Once you have opened up the sketch in the Arduino IDE successfully, click on the tab labeled config.h.

Fill in your `IO_USERNAME` and `IO_KEY` with your Adafruit IO Username and Key. If you're not sure where to find them, there's a section about doing that on the Stream Deck Plugin page.

Fill in the `WIFI_SSID` and `WIFI_PASS` with your WiFi name and password.

Make sure `#define USE_AIRLIFT` is uncommented.

Go back to the MessagePanel tab and if you decided to use a different feed name, find the following line and change it.

```c
AdafruitIO_Feed *counter = io.feed("messagepanel");
```

Be sure the correct board (Adafruit Metro M4 Airlift Lite) and port are selected and upload the sketch to your Metro M4 Express Airlift.

3D Printing

To make this project look nice and complete, I designed a backing to the RGB Matrix Panel that nicely houses the board and attaches to the back of the Matrix Panel. This was designed for the 64x32 RGB LED Matrix with 4mm Pitch.
Also, Adafruit switched over to carrying a 32x64 Matrix with a different set of mounting holes and there is a version of the design that has been updated to accommodate the newer matrix. So now there are four versions of the printable backing.

**Single Piece for Newer Matrix**

For larger volume printers such as the Creality CR-10 and Artillery Sidewinder X1, you can print the backing as a single piece. You can download the STL or 3MF files here:

- Download STL for Single Piece for the Newer Matrix
  
  https://adafru.it/YbQ

- Download 3MF for Single Piece for the Newer Matrix
  
  https://adafru.it/YbR

**Single Piece for Older Matrix**

If you have the older matrix, you can still download the STL or 3MF files here:

- Download STL for Single Piece for the Older Matrix
  
  https://adafru.it/XEt
Two Pieces for the Newer Matrix

Due to the size of the message panel, the design is also available as a 2-part print so that it can be printed on a standard 3D Printer.

You can download the STL files here:

Download STL for the Newer Matrix Board Half
https://adafru.it/YbT

Download STL for the Newer Matrix Other Half
https://adafru.it/YbU

You can download the 3MF files here:

Download 3MF for the Newer Matrix Board Half
https://adafru.it/YbX
Two Pieces for the Older Matrix

If you have the older matrix, you can still download the STL files here:

Download STL for the Older Matrix Board Half
https://adafru.it/XEu

Download STL for the Older Matrix Other Half
https://adafru.it/XEv

You can download the 3MF files here:

Download 3MF for the Older Matrix Board Half
https://adafru.it/YbY

Download 3MF for the Older Matrix Other Half
https://adafru.it/YbZ
This backing is designed to allow the entire project to be mounted onto a wall. The tabs hold it together well enough for mounting purposes. For mounting the Metro M4 WiFi, you will need these heat set inserts which it was designed for.

Brass Heat-Set Inserts for Plastic - M3 x 3mm - 50 pack
Wanna improve the connection strength between your project's 3D-printed parts, and also have nice clean surfaces? Instead of gluing bits together, or screwing plastic screws...
https://www.adafruit.com/product/4256

We would also strongly recommend our heat-set insert soldering iron adapter.

Heat-Set Insert For Soldering Irons - #4-40 / M3 Inserts
Wanna improve the connection strength between your project's 3D-printed parts, and also have nice clean surfaces? Instead of gluing bits together, or screwing plastic screws...
https://www.adafruit.com/product/4239
If you find yourself doing a lot of heat-set inserts, you could even build a heat-set insert jig by following our Heat Set Insert Rig (https:// adafruit.it/Fis) guide.

3D Printer Settings

This model worked with printing using the following Slicer settings:

- Use supports for the side with tabs if printing a 2-piece model
- 10% Infill
- 0.20mm Layer Height

Assembly

Assembly of this project is pretty straightforward. Most of special instructions revolve around the RGB shield assembly. Because the Metro M4 Express Airlift is still a Metro M4, you will need to cut one of the traces and add a wire.

You may want to consider reversing the direction of the screw terminal blocks so the wire isn’t up against the wall of the mount.

You can follow assembly instructions for the RGB shield in our 32x16 and 32x32 RGB LED Matrix (https://adafruit.it/FBU) guide.

To assemble, you will want to start by adding the heat-set inserts into place where the Metro M4 Express Airlift will be mounted.

Carefully inset the heat-set inserts using a heat-set tool or jig.
Carefully line up the Metro M4 Airlift onto the heat-set inserts. Make sure the power jack is able to go through the hole. It's a little tight, so you may need to use a sharp knife to carve away any excess filament.

Install the four screws to secure the Metro M4 Airlift.
Next, snap the two halves together.

Install the four magnetic screws into the short ends of the RGB Panel.

Once you have your RGB Shield soldered up, connect the power wiring harness to the screw terminals on the RGB Shield.
Place the shield onto the Adafruit Metro M4 Express Airlift.

Connect the ribbon cable that came with the panel between the Panel and the shield.

Next, using a level, mount it onto the wall using some long wood screws. Do not over-tighten.
Attach your ribbon cable and one of the power connectors to your RGB Panel.

With all the wires connected, carefully place any extra long wires into the side without the board.

The RGB Matrix Panel works by using the special magnetic screws that come with the panel to magnetically stick to the wall mounting screws. You may need to loosen the wall mount screws a bit if it's not sticking well in all four corners.
Once that is done, connect power. It should say connecting... followed by the last message that was stored in your feed value.

Troubleshooting

The screen is not working correctly

Make sure to cut the trace on the shield next to CLK and run a wire over to A4. For more information, see Connecting Using RGB Matrix Shield (https://adafruit.it/FBU) and scroll down to the Metro M4 Usage section.

There is still a bunch of random pixels on the screen and it's unreadable

Try connecting power only or try a different ribbon cable.

The display keeps fading out or resetting.

Make sure you have enough external power going to the Metro M4 Express Airlift. The RGB Matrix requires a lot of power.