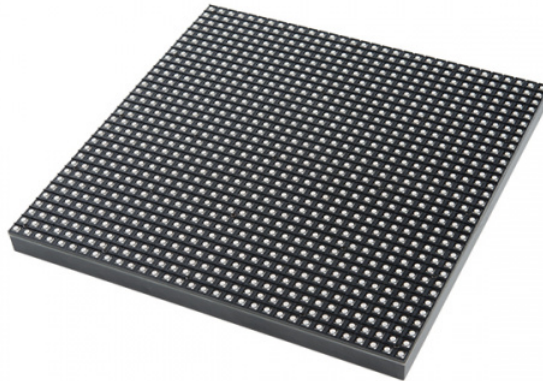


RGB Panel Hookup Guide

Introduction

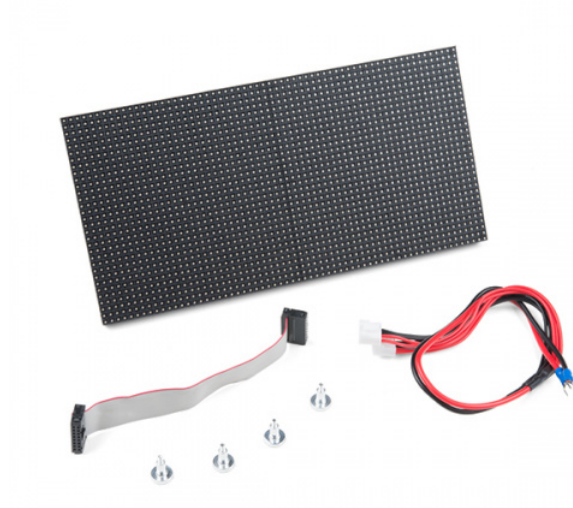
Are you looking to add a *lot* of color to your project? These *massive* RGB LED panels are an awesome place to start. You can create animations, games, or all sorts of other fun displays with them. Depending on the manufacturer, these panels can come in different sizes, LED pitch, and scan rates. Here are the ones that SparkFun currently carries in the catalog:

- 1024 pixels (3072 total LEDs!) 32x32 pixel panel with 1:16 scan rate measuring 7.5"x7.5"
- 2048 pixels (6144 total LEDs!) 32x64 pixel panel with 1:16 scan rate measuring 5x10"



RGB LED Matrix Panel - 32x32

● COM-14646



RGB LED Matrix Panel - 32x64

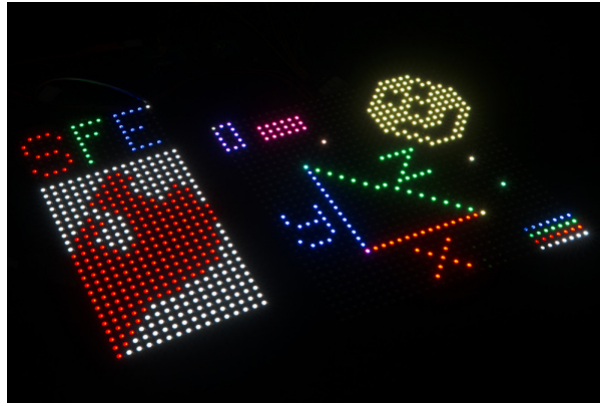
● COM-14718

Note: The 16x32 version has been retired from our catalog, but we are keeping the information in the tutorial for reference.



RGB LED Panel - 16x32

In this tutorial we'll show you just how, exactly, these panels operate. We'll dig into the hardware hookup and examine how to best power them. Then we'll work up a demo sketch and control them with Arduino.



A 16x32 RGB LED panel to the left, and a 32x32 panel to the right.

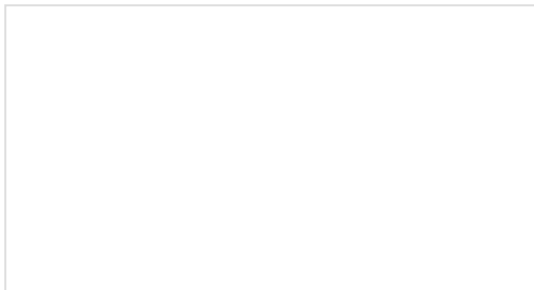
Required Materials

On top of either size panel, you'll also need:

- At least an Arduino Uno (or comparable ATmega328-based Arduino). These panels really stretch the Arduino to its limits. If you have an Arduino Mega 2560 you may want to whip that out instead. Any size higher than a 32x32 panel requires an Arduino Mega 2560 or faster microcontroller.
- Two packs of male-to-male jumper wires. You'll need around 16 to wire from the panel to your Arduino.
- **A 5V power supply.** You'll need something that can source a high amount of current. A simple 5V (1A) wall adapter does work, at least in the short run, but you may want to step up to a higher capacity supply, like the 12V/5V (2A) or 5V/2A wall adapter.
- You'll also need some method to connect your power supply to the panel. The panel includes a 4-pin polarized connector and spade-terminated cable for its power supply. Check out the next page for help finding a power source and cable.

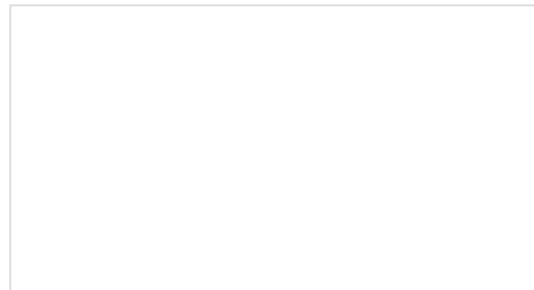
Suggested Reading

Before following along with this tutorial, we recommend reading through these tutorials first:



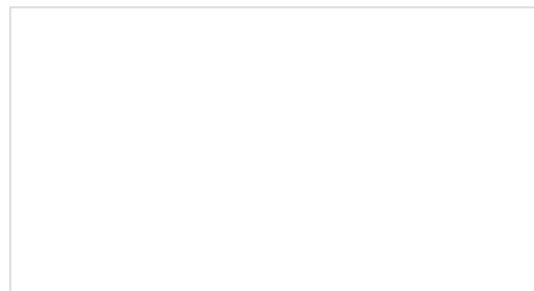
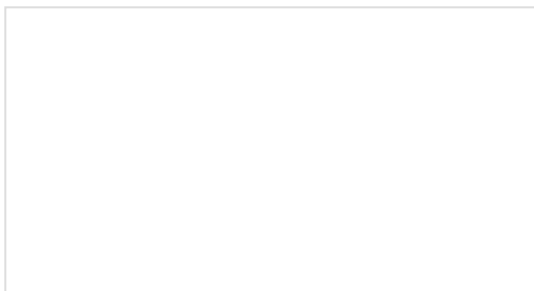
How to Power a Project

A tutorial to help figure out the power requirements of your project.



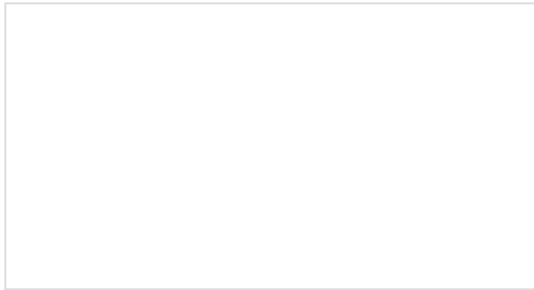
Working with Wire

How to strip, crimp and work with wire.



What is an Arduino?

What is this 'Arduino' thing anyway?

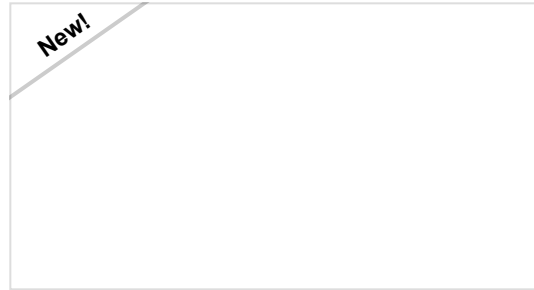


Light-Emitting Diodes (LEDs)

Learn the basics about LEDs as well as some more advanced topics to help you calculate requirements for projects containing many LEDs.

Shift Registers

An introduction to shift registers and potential uses.



12V/5V Power Supply Hookup Guide

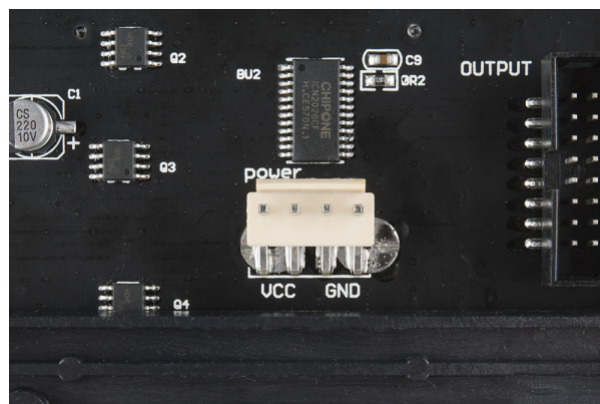
In this tutorial, we will replace the 12V/5V (2A) power supply's molex connector with two male barrel jacks adapters.

Powering the Panel

Power Connector

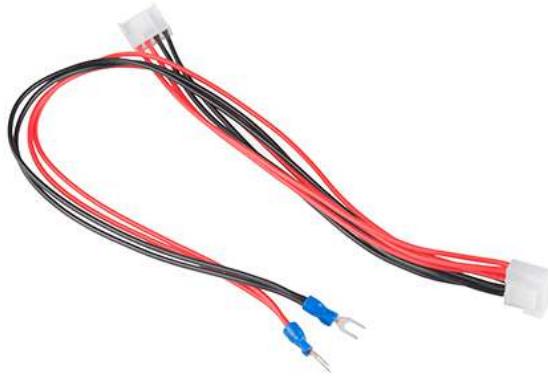
⚠ Warning! Don't use the 5V supply from your Arduino. Those are only spec'ed to supply about 800mA, and the Arduino's already eating into that capacity a bit.

These panels require a **regulated 3.3-5V** supply for power. And that supply needs to be able to source a good amount of current – **up to 2A in the worst case** (all pixels bright, hot, white). For a 32x64, one of these panels was pulling about $\sim 3.36A-3.43A$ *without a heat sink* – **so about up to 4A worst case**. A 4-pin (2 for VCC, 2 for GND), 0.15"-pitch polarized connector should be used to supply power to the panel. Depending on the manufacturer, the color and location of the power connector may be different.



Power Cable

Included with the panel is a dedicated cable for power. It's a 0.15" pitch 4-pin polarized connector. The included cable is terminated with both a female polarized connector, and a pair of spade terminals.



Here are a few methods we've used to power the panel:

Longer-Term: 12V/5V Power Supply

This is our recommended method. Combine:

1. A 12V/5V 2A Power Supply, which should be enough to keep the display running. (Just don't hook up the 12V output to it!)
2. An IEC C13 Cable to connect AC power to the supply.
3. A 4-pin Molex Connector w/ Pigtail to interface the supply to panel.



The ingredients for our power supply and cable.

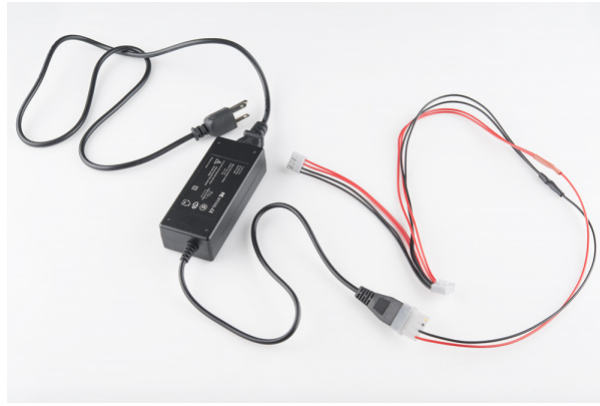
To begin, we snipped the spade connectors off of the panel power supply cable. And then stripped the newly unterminated ends.

Then we spliced the Molex pigtail to the LED panel's power cable by connecting the **red wires together**. Do the same for the black wires (make sure you use the black wire next to the red on the Molex pigtail). Make sure you are connecting to the 5V and GND pins and **NOT** the 12V pin. Before connecting to the RGB Matrix Panel, test the connection with a multimeter.



Spliced Wires

Finally, cover the splice with heat shrink or electrical tape, and voila! That's a beautiful power cable.



Finished panel power supply cable.

This is a nice, sturdy interface between the panel and a solid power supply. If you're looking for something easier, but less reliable check the below option.

Long-Term: Mean Well Switching Power Supply

For those that want to push the panels to the limit (i.e. setting the pixels on a 32x64 panel at full white at maximum capacity), combine:

1. A 5VDC/20A Mean Well switching power supply - which is more than enough for your panel.
2. A wall adapter cable (North America or European standard) depending on your country.



Short-Term: Barrel Jack

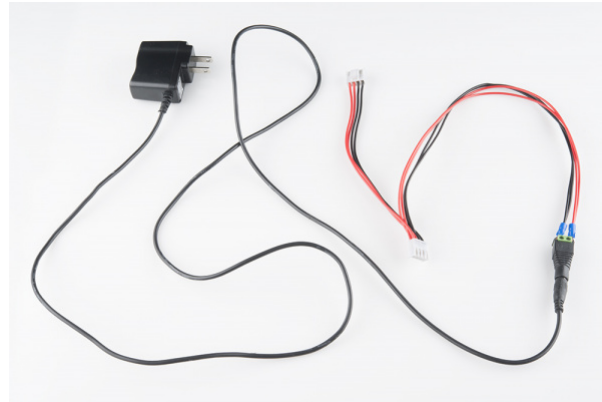
Grab a 5V wall adapter. Both been tested to work with the panels as well. At least in the short term.

- 5V/2A Wall Adapter
- USB 5V/1A Wall Adapter (with USB Barrel Jack Adapter)

Use the power supply in conjunction with a female barrel jack adapter and screwdriver to get a quick and dirty connection between the spade and barrel jack.

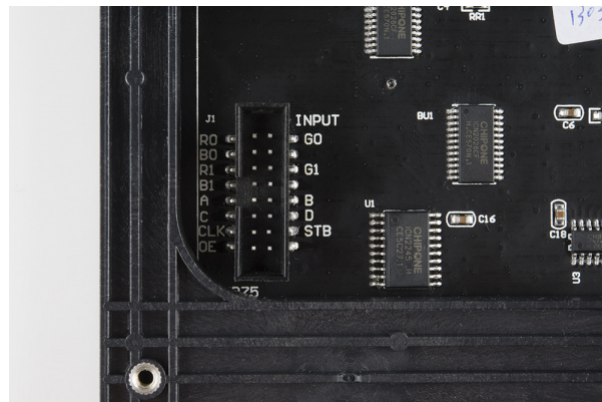


The final connection should look like the image below.



Hardware Hookup

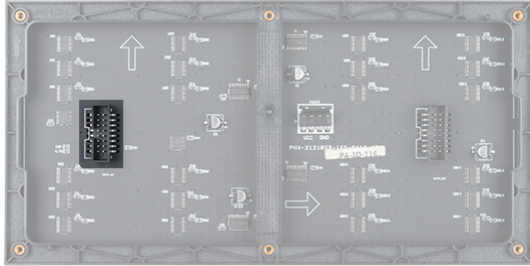
Before we can get into the code portion, there's quite a bit of wiring to do. The RGB panels have a pair of 16-pin (2x8) IDC connectors, and we need to wire up to most of those pins. Conveniently, both panels have the connector pins labeled (the unlabeled pins are ground). As we're hooking up to the panel, make sure you use the connector labeled *Input*. The labeling may be slightly different depending on the manufacturer.



Connector labels on a 32x32 panel. 16x32 has the same layout, except that D is a no connect (NC) instead. 32x64 has the same layout but no labels.

Note to 16x32 Panel Users: The 32x16 panel has the exact same pinout as the 32x32, except there is no "D" pin. Instead of "D", that pin (12 on the connector) is a no connect (NC), you can leave it alone.

Labels on 32x64 and 16x32 Panel Users: On top of that, some panels do not have labels on the connector pins -- instead there is an **arrow (◀) indicating pin 1**, in the top-left corner of the connector (it's obscured by the frame, but visible if you peek in at the right angle). That pin 1 arrow indicator points to the "R0" pin, and the pinout follows that of the 32x32 panel from there. We have also seen panels come with their pins label **G1, G2, R1, R2, B1 & B2** instead of **G0, G1, R0, R1, B0 & B1**. The wiring will be the same for both cases by connecting to the connector on the left.



Hookup Table

Here are the pin connections between LED panel connector and Arduino:

Panel Pin Label	Cable Connector Pin #	Arduino Uno (Atmega328P) Pin	Arduino Mega 2560 Pin	Notes
R0	1	2	24	Red Data <i>(columns 1-16)</i>
G0	2	3	25	Green Data <i>(columns 1-16)</i>
B0	3	4	26	Blue Data <i>(columns 1-16)</i>
GND	4	GND		Ground
R1	5	5	27	Red Data <i>(columns 17-32)</i>
G1	6	6	28	Green Data <i>(columns 17-32)</i>
B1	7	7	29	Blue Data <i>(columns 17-32)</i>
GND	8	GND		Ground
A	9	A0		Demux Input A0
B	10	A1		Demux Input A1

C	11	A2	Demux Input A2
D	12	A3	Demux Input E1, E3 (32x32 panels only)
CLK	13	11	LED Drivers' Clock
STB	14	10	LED Drivers' Latch
OE	15	9	LED Drivers' Output Enable
GND	16	GND	Ground

For a more step-by-step approach, follow along below. We use male-to-male premium jumper wires to wire between the included ribbon cable and our Arduino.

Connect Data Pins: R0, B0, G0, R1, G1, and B1

These LED driver (shift register) data pins are hard-coded in the Arduino library and can't be moved. As listed in the table above, R0, G0, B0, R1, G1, and B1 go to the Arduino Uno's **pins 2 through 7** respectively. If you're wiring the panel up to an Arduino Mega, these pins should be connected to **pins 24-29** respectively.

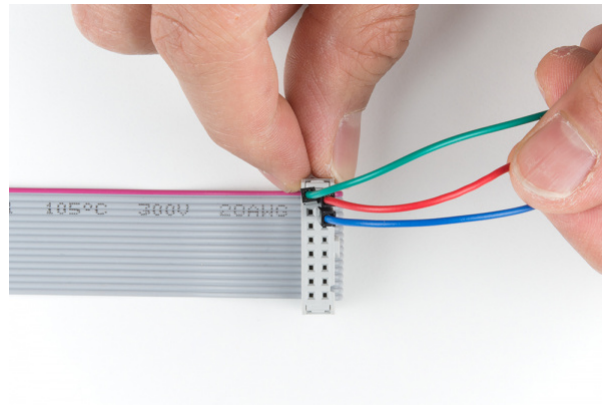
For reference when wiring the pins, try looking down at the IDC connector with red wire on top. The cable connector's pin 1 is relative to the top right. If you look really closely at the molding, you may also arrow (◀) pointing to that pin. Also, note that the tab for polarity is on the right side on either end of the cable.



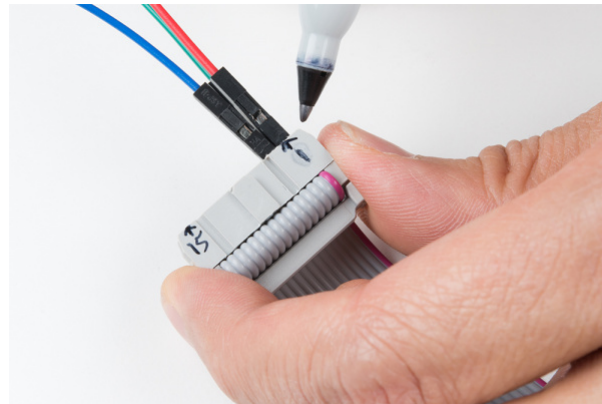
IDC Connector Highlighted w/ Arrow Pointing to Pin 1

Tip: The **odd** numbered pins will be along the right side with the polarity marker and the tab. The pin holes on the left will be even numbered. Depending on the manufacturer, you may have received a black connector instead of a grey.

Once you decide what side to connect, start wiring pin 1 by connecting a red wire for R0. Then connect pin 2 by connecting a green wire for G0. After connecting pin 3 using a blue wire for B0, continue wiring the pins based on the hookup table.



To help keep track of what side you are connecting to, feel free to label your connections with a marker.



Connecting the Clock Pin

This is the last pin that has some restriction on where it can go – it must be connected to one of Arduino's **port B** pins. That means it must be either *8, 9, 10, 11, 12, or 13*. The example code has it defined as **pin 11** in the hookup table. Make sure to check your pin definition for the clock as it may vary with the code you are using.

Connecting to A, B, C, (D for 32x32 users), OE, and STB Pins

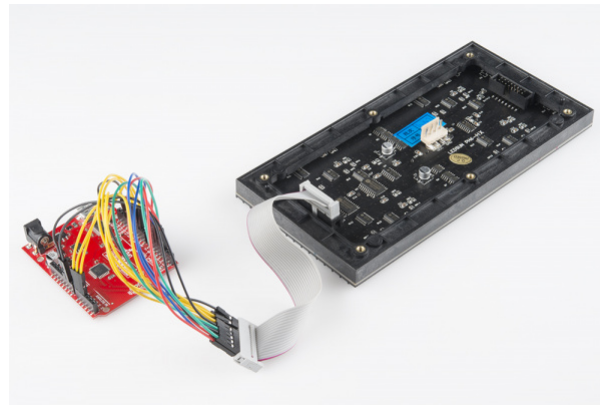
These five (or six if you're using a 32x32 matrix) pins can be plugged in anywhere you may have space on your Arduino. Although, there's probably not a lot of room left... We chose to stick A, B, and C in **pins A0, A1, and A2** respectively. OE connects to **pin 9**. STB to **pin 10**. And, if you're using a 32x32 matrix, D goes to **A3** as stated in the hookup table.

Feel free to swap those up if your application requires. Just make sure you switch it up in the example code too.

Connecting to Ground(s) Pins

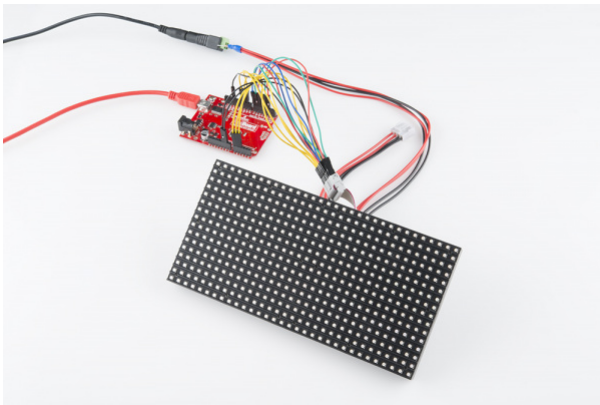
Last, but certainly not least (well, maybe, if we're talking about potential) is ground. There are three unlabeled pins on the connector which should all be tied to ground.

If you don't have anything else plugged into them, there should be three ground pins available on your Arduino. If you're struggling to find ground pins, though, you should be able to get away with only plugging one of the ground pins to your Arduino. Woo color coded wires!

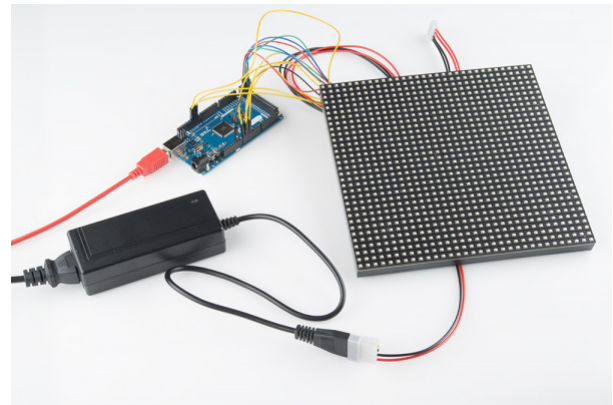


Power

Once you are finished connecting the LED panel to an Arduino, add your 5V power supply to the panel's power connector. Don't forget to add power to your Arduino!



16x32 Fully Connected to the RedBoard

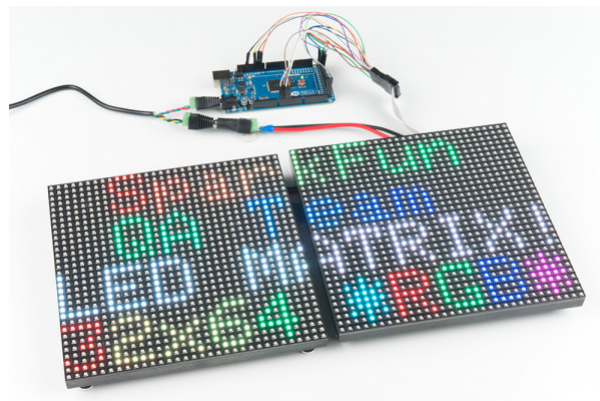


32x32 Fully Connected to the Arduino Mega 2560

Connecting to 32x64 RGB LED Panels

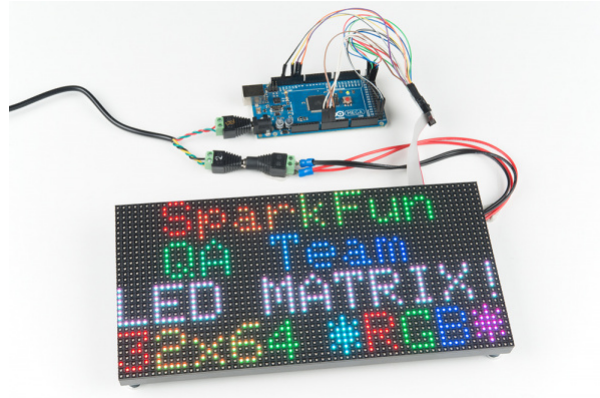
For anyone connecting to the 32x64 RGB Panels, you will **need** to use an Arduino Mega 2560 with the library. Otherwise, the panels will not be able to display as expected due to the limitations of the library and the Arduino Uno.

To daisy chain two 32x32 RGB matrices together, connect another IDC cable from the output of the first panel to the input of the second panel. Then connect the second 4-pin polarized connector to the input power connector. After modifying the `test_shapes_32x64.ino`, the display will output as a 32x64 matrix as shown below.



Heads up! When daisy chaining the RGB LED matrix panels, make sure that they have the same scan rate!

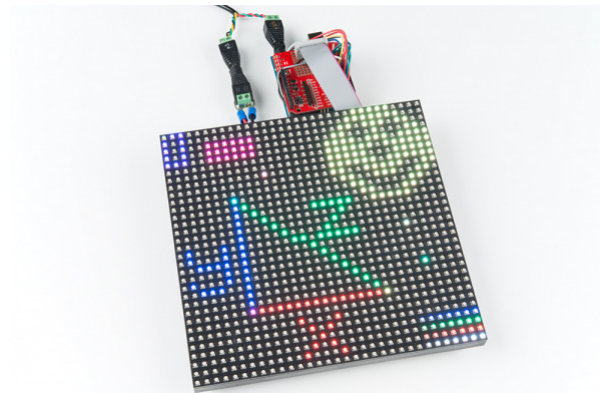
Here's an example with the 32x64 matrix with 4mm pitch.



When using an array higher than 32x64, another library would be a better option with a Teensy 3.0+, FPGA, or Raspberry Pi. You may need to level shift to convert 3.3V to 5V for the RGB LED panel to recognize the I/O signals. For more information, check out the links in the Resources and Going Further.

Custom Shield Adapter

If you have a prototyping shield, try making a custom shield adapter for a more secure connection. Here's an example of using an XBee shield's prototyping area with the connection specifically for the RGB LED panels. Follow the steps outlined above but solder the connection together.



Magnetic Mounts

Depending on the supplier, you may receive a set of magnetic mounts. Add it to the panel's mounting holes to stick on a fridge or metal wall! They also work great as a standoff. Just make sure to secure and insulate your wires to prevent any shorts if there is metal behind the panel.



Arduino Library Installation

Note: This example assumes you are using the latest version of the Arduino IDE on your desktop. If this is your first time using Arduino, please review our tutorial on installing the Arduino IDE. If you have not previously installed an Arduino library, please check out our installation guide.

Compatibility: The 16x32 works at a 1:8 scan rate. The 32x32 and 32x64 examples work with a 1:16 scan rate. Depending on the manufacturer, there may be different scan rates for the the LED Matrix Panels. Using different scan rates with the example code may cause unexpected behaviors.

Our example code is going to make use of Adafruit's most excellent **RGBMatrixPanel** library, which also requires their **AdafruitGFXLibrary**. You can obtain these libraries through the Arduino Library Manager by searching for those names. Or you can manually install them can by grabbing both of them [RGBMatrixPanel and AdafruitGFXLibrary] from their GitHub repositories.

For your convenience, we packaged those libraries with the serial paint example code used in the tutorial:

SERIAL PAINT EXAMPLE CODE (ZIP)

Library Examples

The **RGBMatrixPanel** library includes a number of fun examples to help show how the library can be used. They're awesome. Check them out under the **File > Examples > RGBMatrixPanel** menu in Arduino. (Definitely check out the *Plasma_16x32* or *Plasma_32x32* examples!). Make sure to adjust the code based on your hardware hookup. In this tutorial, we physically connected the clock pin to 11. Therefore, you need to adjust the defined CLK pin from

```
// If your 32x32 matrix has the SINGLE HEADER input,  
// use this pinout:  
#define CLK 8 // MUST be on PORTB! (Use pin 11 on Mega)  
#define OE 9  
#define LAT 10  
#define A A0  
#define B A1  
#define C A2  
#define D A3
```

to:

```
// If your 32x32 matrix has the SINGLE HEADER input,  
// use this pinout:  
#define CLK 11 // MUST be on PORTB! (Use pin 11 on Mega) <---CHANGE!  
#define OE 9  
#define LAT 10  
#define A A0  
#define B A1  
#define C A2  
#define D A3
```

32x32 Examples for 32x64: You can also use the "..._32x32.ino" examples in the **RGBmatrixPanel** library by adjusting the code for the 32x64 RGB matrix panels (i.e. **colorwheel_32x32**, **plasma_32x32**, etc). Just add the number " 64 " as a parameter when creating an instance of the RGBmatrixPanel class:

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

Example Code

Serial Paint

We wanted to write another fun sketch that provided an interactive way to explore with the panels and the Arduino library. What we came up with is a serial-controlled paint program. With this sketch, you can use the serial monitor (or, better yet, another terminal program) to control a cursor and draw on the matrix.

Download and unzip the sketch included in the zipped libraries using the link below if you have not already.

[SERIAL PAINT EXAMPLE CODE \(ZIP\)](#)

- **Selecting Your LED Matrix Size**

Before uploading, make sure the sketch is set up to work with your panel where it says:

```

/* - One of these should be commented out!
   - Also, make sure to adjust the saved image in the <bitmap.h> file.*/

/* ===== For 32x64 LED panels: =====
   You MUST use an Arduino Mega2560 with 32x64 size RGB Panel */
//RGBmatrixPanel matrix(A, B, C, D, CLK, LAT, OE, false, 64); // 32x64

/* ===== For 32x32 LED panels: ===== */
RGBmatrixPanel matrix(A, B, C, D, CLK, LAT, OE, false); // 32x32

/* ===== For 32x16 LED panels: ===== */
//RGBmatrixPanel matrix(A, B, C, CLK, LAT, OE, false); // 32x16

```

By default, the Serial Paint example uses the 32x32. If you are using 32x64, make sure to uncomment the following line by removing the “ // ”

```
//RGBmatrixPanel matrix(A, B, C, D, CLK, LAT, OE, false, 64); // 32x64
```

Then comment out the following line by adding a “ // ” at the beginning of the line.

```
RGBmatrixPanel matrix(A, B, C, D, CLK, LAT, OE, false); // 32x32
```

• Selecting the Saved Bitmap Image

Make sure to also uncomment/comment out the `bmp[]` array in the **bitmap.h** file by removing and adding “ /* ” and “ */ ” for your respective size.

• Upload!

Once you have adjusted the code to your screen, upload it to the Arduino! After upload, a single pixel should be blinking at the top left of the panel. It doesn't look like much, but that's a good sign.

Using the Sketch

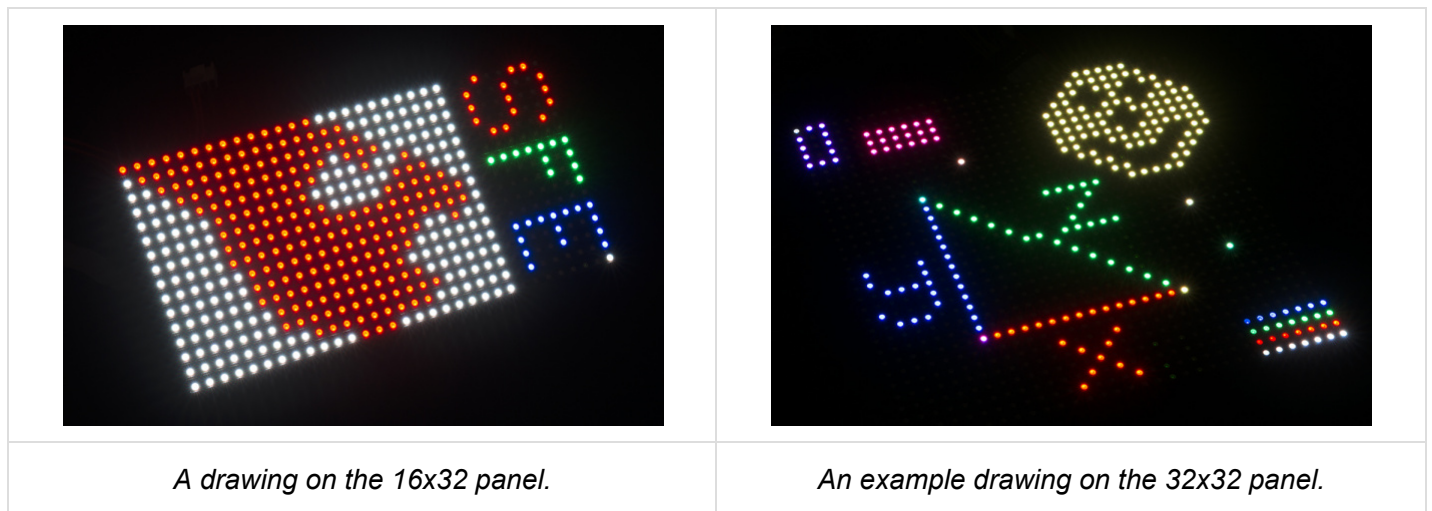
To control the program, open up your serial terminal to **9600 bps**. Try hitting sending `l` (lowercase 'L') through the serial monitor, which should load the demo bitmap. You can send `E` (uppercase) to erase the screen.

The idea of this sketch is: move the cursor around to draw pixels, shapes, or text. Here are the commands made available by the sketch (they are case-sensitive):

- **Movement:** `w , a , s , d` (up, down, left, right)
- **Draw Pixel:** `Spacebar`
- **Erase Pixel:** `e`
- **Erase Screen:** `E`
- **Fill screen** with active color: `f`
- **Color Control:**
 - **Red value up:** `R` (values between 0 [off] and 7 [most bright])
 - **Red value down:** `r`
 - **Green up/down:** `G / g`
 - **Blue up/down:** `B / b`
 - **Copy color:** `z` (copies a color under the cursor)
- **Shape Drawing:**
 - **Line:** press `v` to place starting point. Then move cursor to endpoint and press `v` **again**.

- **Rectangle:** press `x` or `X` to place first corner. Then move your cursor to where you want the diagonal corner. Then press either `x` for an **empty** box, or `X` for a **filled** box.
- **Circle:** press `c` or `C` to place the *center* of the circle. Then move your cursor to where you want the outside edge of your circle to be. Then press `c` for an **empty** circle or `C` for a **filled** circle.
- **Text:** press `t` to go to text mode. Now **any characters** received will be displayed on the panel. It'll wrap around from one line to the next, but not from bottom to top. Press ``` (above Tab / left of 1) to exit text mode.
- **Print:** press `p` to print an array of your drawing to the serial terminal. You can copy this, and put it back in your sketch if you want to load it again.
- **Load:** press `l` to load a pre-defined array from the sketch. The sketch includes a demo array, which was created from the print command. Follow this example to load your own drawings!

Give the paint sketch a try! See if you can make the next great ~~Lite-Brite~~ LED Panel picture. If you make something neat, share it with us! Don't laugh. I drew that SFE flame one pixel at a time! Here are our creations:



Try modifying the code to change the color and move around the matrix using potentiometers and buttons. You can see it in action in our product showcase:

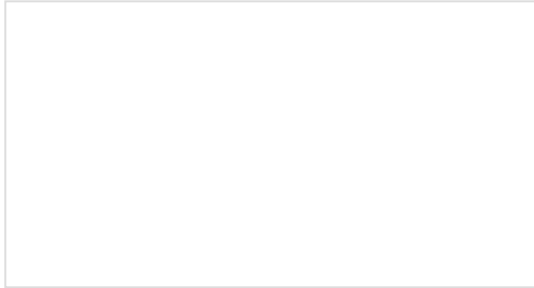
SparkFun 12-13-13 Product Showcase: Etch-A-Sketch



Resources and Going Further

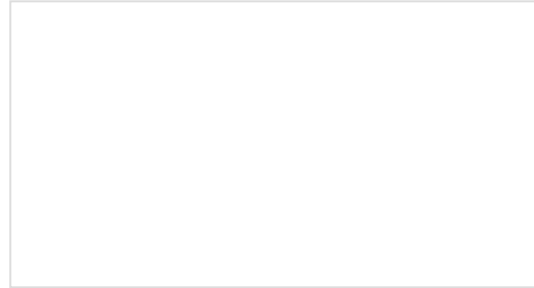
Now that you know how to make use of these beautiful RGB LED matrices, what nifty project are you going to create with them? Here are some more resources, if you need them:

- GitHub Repos
 - Adafruit's RGBMatrixPanel Library – Arduino example library used in this tutorial.
 - RPi RGB LED Matrix – Another example using a Raspberry Pi.
 - SmartMatrix – Another example using a Teensy.
- Ray's Logic Reverse Engineering Page
- SparkFun Product Showcase: Etch-A-Sketch
- Product Showcase: RGB LED Matrix Panel - Demo of the 32x64 RGB LED Panel
 - GitHub Gist Product Demo - Demo for the 32x64 product showcase.



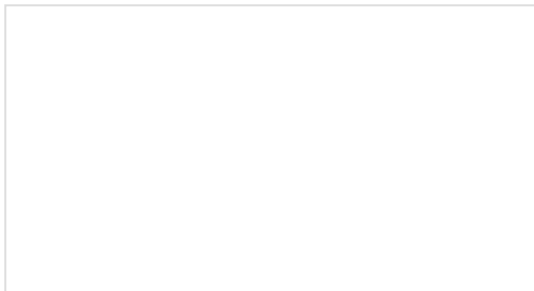
RGB Panel Jumbotron

This tutorial will show you how to combine a webcam, a 32x32 RGB LED panel, and a Teensy 3.1 to stream video from the webcam, pixelate it, and display it on the LED panel - LIVE.



Hackers in Residence: The Sound Visualizer

A fun project that uses a Raspberry Pi and a custom Java app to create your own sound visualizer using a RGB LED matrix.



Hackers in Residence: The Sound Visualizer Pt. 2

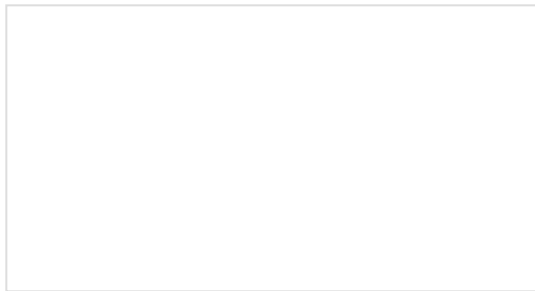
An addition to a previous project, this time using a PC and a custom Java app to create your own music visualizer using a RGB LED matrix.

Maybe try making a hover pong game with the 32x32 RGB LED Matrix using a gesture sensor. The example code can be found in the GitHub repo HoverPong

SparkFun ZX Distance Sensor

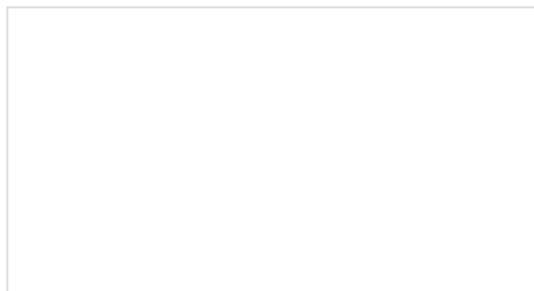


If you're looking for some inspiration or more stuff to learn, check out these tutorials:



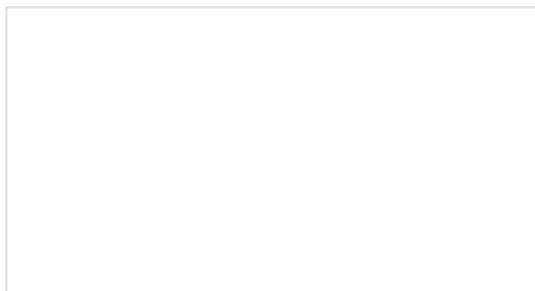
Light

Light is a useful tool for the electrical engineer. Understanding how light relates to electronics is a fundamental skill for many projects.



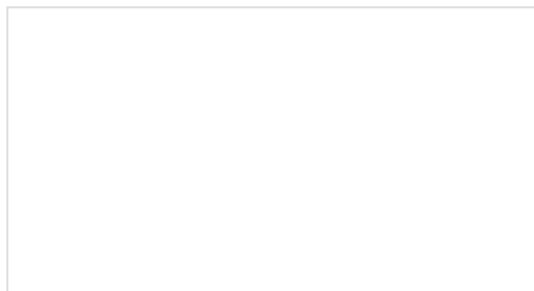
Serial Graphic LCD Hookup

Learn how to use the Serial Graphic LCD.



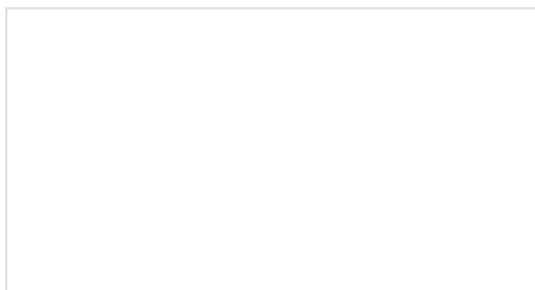
LED Light Bar Hookup

A quick overview of SparkFun's LED light bars, and some examples to show how to hook them up.



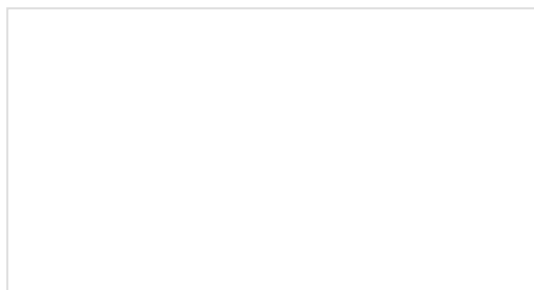
Graphic LCD Hookup Guide

How to add some flashy graphics to your project with a 84x48 monochrome graphic LCD.



SparkFun LED Array (8x7) Hookup Guide

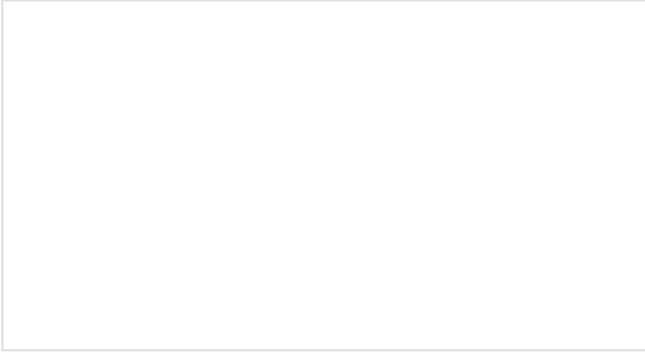
Getting started with the Charlieplexed 8x7 LED array.



Marquee Party Bag

This tutorial provides everything you need to know to make your own Marquee Party Bag!

Or check out this blog posts for ideas. Remember, since the 1:8 scan rate 32x32 RGB LED Panel is a SparkX product, support is limited. There is a useful modified Arduino library specifically for a panel with 1:8 scan rate.



Everything You Didn't Want to Know About RGB Matrix Panels

MARCH 29, 2018