Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

adafruit learning system

32x16 and 32x32 RGB LED Matrix

Created by Phillip Burgess

Last updated on 2016-07-18 05:33:29 PM EDT

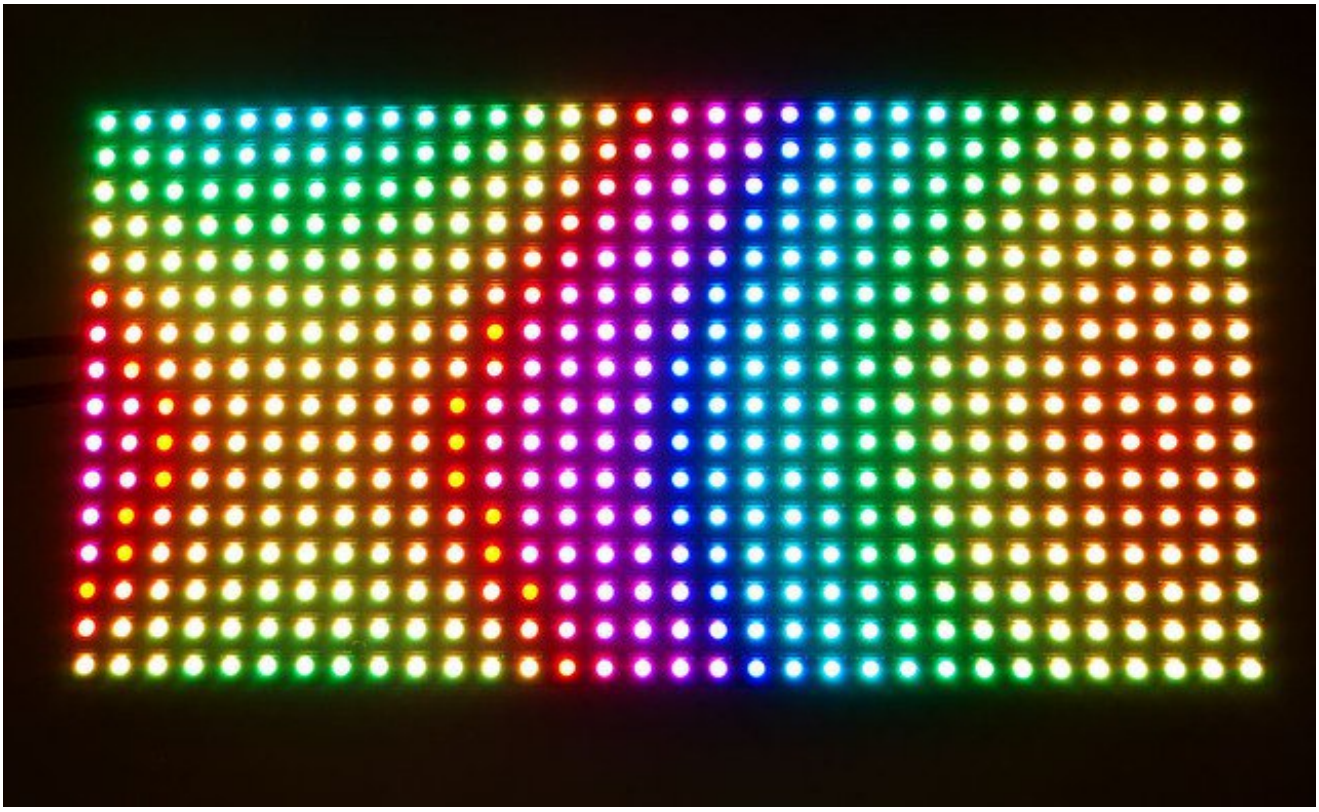
Guide Contents

Overview

Bring a little bit of Times Square into your home with our RGB LED matrix panels. These panels are normally used to make video walls — here in New York we see them on the sides of buses and on bus stops — to display animations or short video clips. We thought they looked really cool so we picked up a few boxes from the factory. One has 512 bright RGB LEDs arranged in a 16x32 grid on the front, the other has 1024 LEDs in a 32x32 grid. On the back is a PCB with IDC connectors (one set for input, one for output: in theory you can chain these together) and 12 16-bit latches that allow you to drive the display with a 1:8 (16x32) or 1:16 (32x32) scan rate.

These panels require 12 or 13 digital pins (6 bit data, 6 or 7 bit control) and a good 5V power supply, at least a couple amps per panel. We suggest our 2A (or larger) regulated 5V adapters and either a terminal block DC jack, or solder a jack from our DC extension cord. Please read the rest of our tutorial for more details!

Keep in mind that these displays are normally designed to be driven by FPGAs or other high speed processors; they do not have built in PWM control of any kind. Instead, you're supposed to redraw the screen over and over to 'manually' PWM the whole thing. On a 16 MHz Arduino Uno, we managed to squeeze 12-bit color (4096 colors) but this display would really shine if driven by an FPGA, CPLD, Propeller, XMOS or other high speed multi-processor controller.



Of course, we wouldn't leave you with a datasheet and a "good luck!" We have a full wiring diagrams and working Arduino library code with examples from drawing pixels, lines, rectangles, circles and text. You'll get your color blasting within the hour! On an Arduino Uno or Mega, you'll need 12 digital pins, and about 800 bytes of RAM to hold the 12-bit color image (double that for the 32x32 matrix).

> The library works ONLY with the Arduino Uno and Mega. Other boards (such as the Arduino Leonardo) ARE NOT SUPPORTED.

Power

Although LEDs are very efficient light sources, get enough of them in one place and the current really adds up.

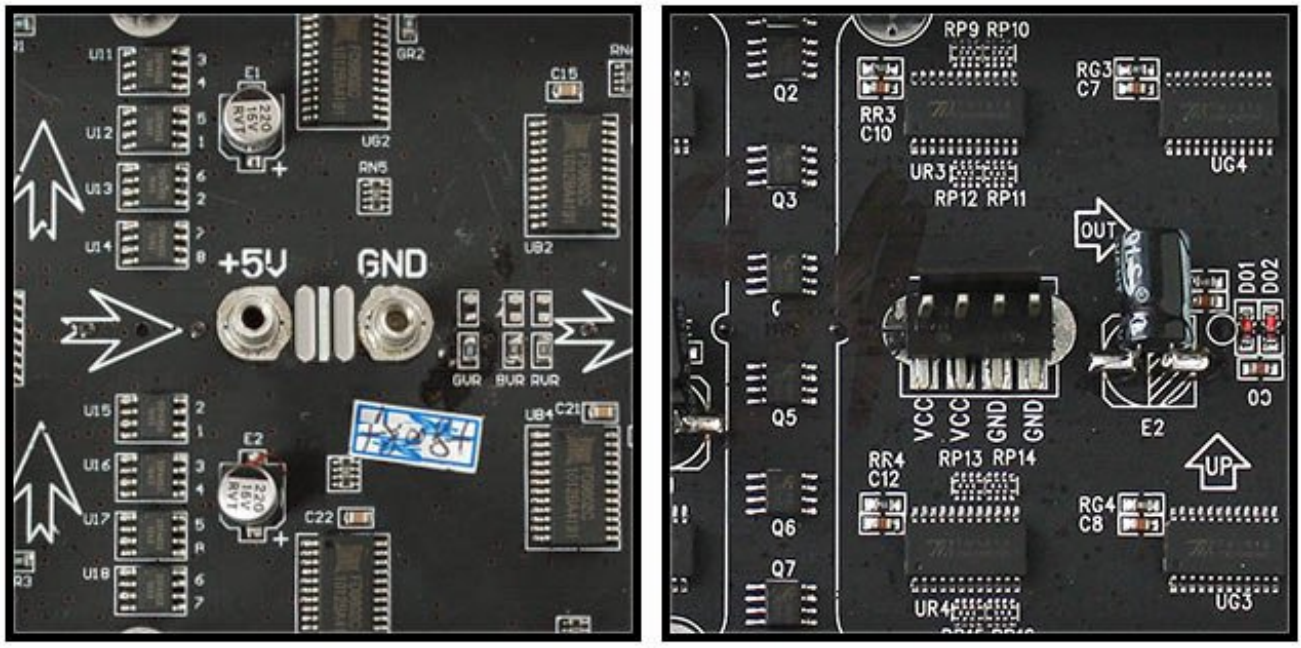
A *single* **32x16** or **32x32** RGB matrix, running full tilt (all pixels set white), can require nearly ***4 Amps*** of current! Double that figure for a **64x32** matrix.

On *average* though, displaying typical graphics and animation, these panels will use less…a **2A** supply is usually sufficient for a single 32x16 or 32x32 panel, or 4A for a 64x32 panel. There's no harm in using a larger power supply rated for more *Amps* (e.g. a 10A supply), but *never* use one with a higher *Voltage* (use **5V, period**)!

On these panels, the power connection is separate from the data connection. Let's begin by connecting a 5V supply…

Our parts suppliers occasionally make revisions to designs. As a result, the connections have changed over time. We'll walk through the different wiring combinations here…pick the explanation that matches the panel(s) you received.

Two different types of power connectors have made an appearance:



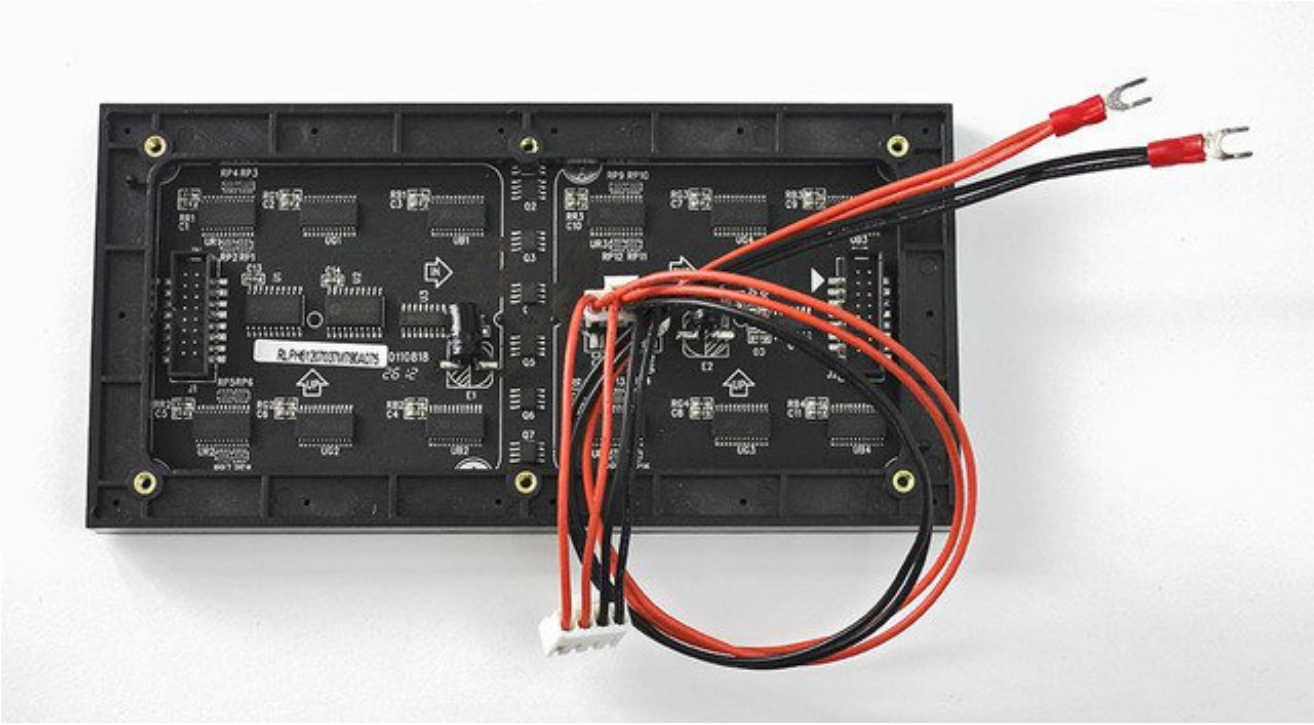
On the left is a screw post power connector (with adjacent pads for soldering wires directly). On the right, a Molex-style header. Some panels will have *two* headers…the power cable included with these panels has connectors for both headers.

With the posts-and-pads connector, you can either screw down the spades from the power cable, or another approach is to cut a 2.1mm jack from this extension cord (http://adafru.it/327) and solder it to

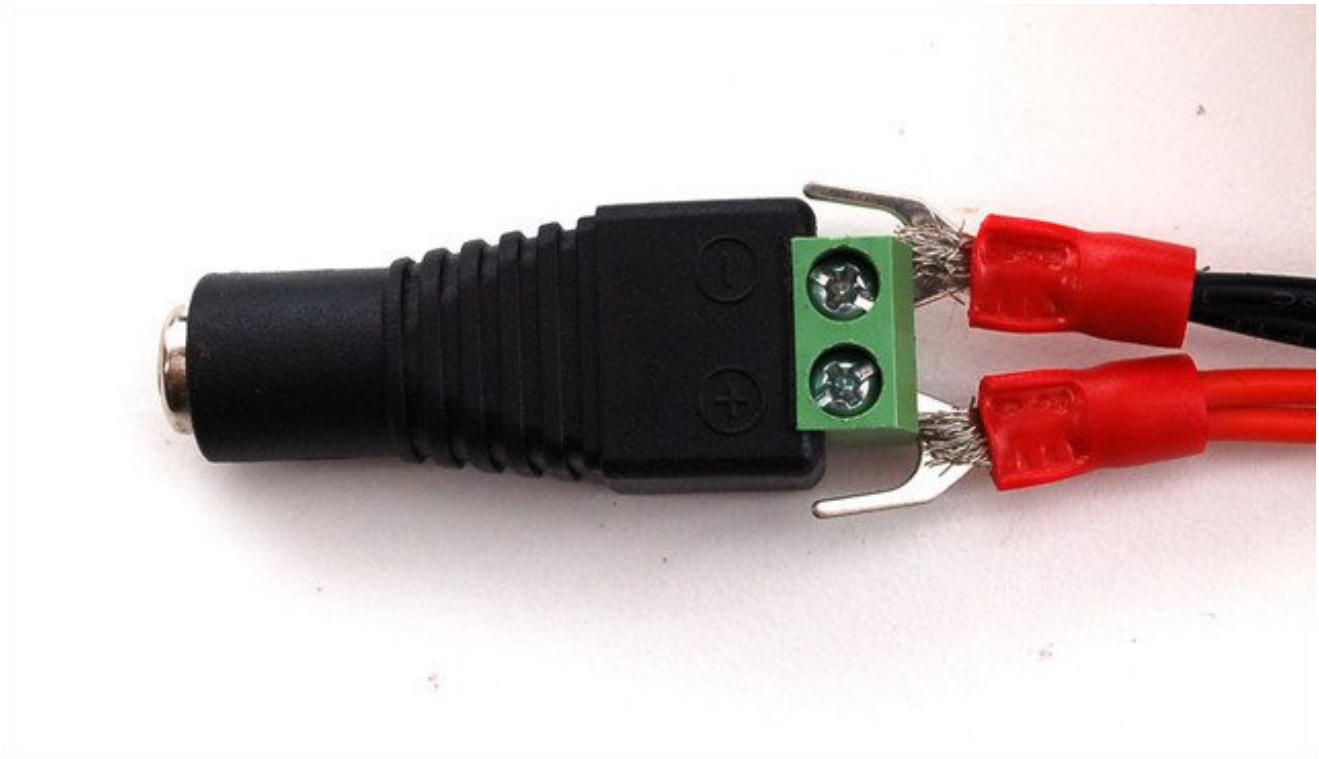
the pads on the panel back. This way you can plug the 5V from a wall adapter (http://adafru.it/276) right in (the one we have in the shop is suggested). Simply cut the other half of the cable off, and strip the wiring so you can solder the red wire to +5 and the black wire to ground.

Solder both pins correctly to the power port. Make sure you get this right because there is no protection diode!

If your panel has the Molex-style header, just plug in the included power cable, observing the correct polarity.



If your power cable came with spades at the opposite end of this power cable, they can be screwed into a 2.1mm terminal block adapter. Works nicely! Don't allow the exposed connectors to contact metal though…you should probably cover this with heat-shrink tube or electrical tape.



You may receive power cables with different endings, e.g. round instead of spade ends, or maybe with another Molex connector. Just strip the cables and wire directly to the power plug

Connections

These panels are normally designed for *chaining* (linking end-to-end into larger displays)…the output of one panel connects to the input of the next, down the line.

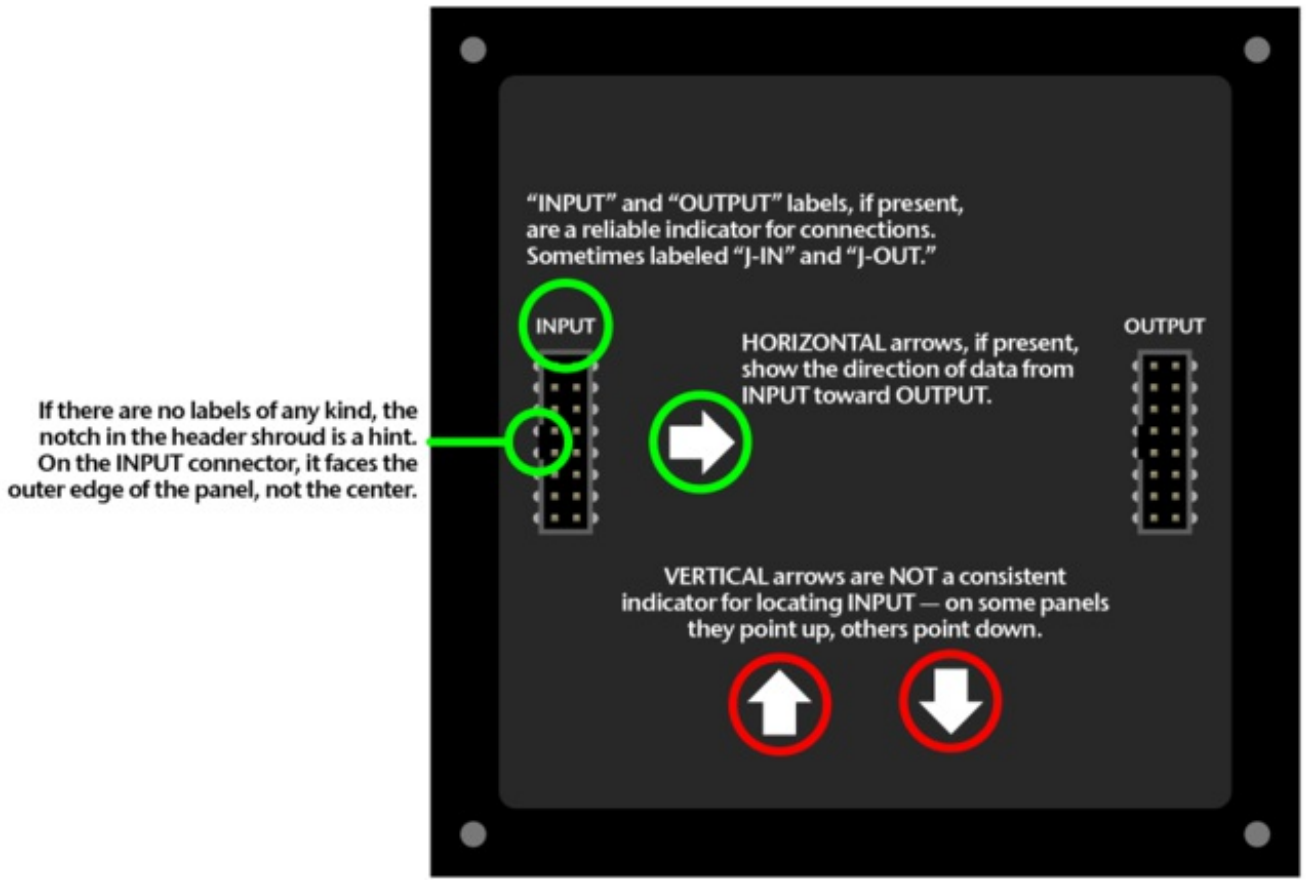
With the limited RAM in an Arduino, chaining is seldom practical. Still, **it's necessary to distinguish the input and output connections** on the panel…it won't respond if we're connected to the wrong socket.

Flip the matrix over so you're looking at the back, holding it with the two sockets situated at the **left and right edges** (not top and bottom).

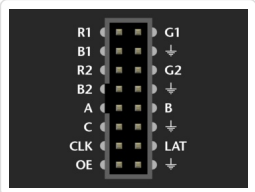
On some panels, if you're lucky, the sockets are labeled INPUT and OUTPUT (sometimes IN and OUT or similar), so it's obvious which is the input socket.

If INPUT is not labeled, look for one or more arrows pointing in the **horizontal** direction (ignore any vertical arrows, whether up or down). The horizontal arrows show the direction data moves from INPUT to OUTPUT — then you know which connector is which.

If no such labels are present, a last option is to examine the plastic shroud around the connector pins. The key (notch) on the INPUT connector will face the outer edge of the panel (not the center).



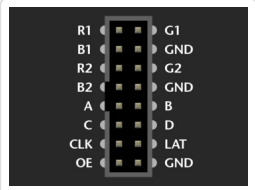
The arrangement of pins on the INPUT connector varies with matrix size and the batch in which it was produced…

- 

  A **32x16** panel uses this pin arrangement. The labels might be slightly different, or the pins might not be labeled at all…but in either case, use this image for reference.

  Notice there are four ground connections. To ensure reliable performance, ***all four*** **should be connected to GND** on the Arduino! A solderless breadboard is handy for making this split.

  Here's the layout for **32x32** and **64x32** panels. We'll call this "**Variant A**." Some panels use different labels, but the functions are identical.

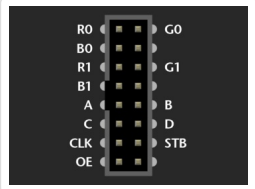
- 

  The layout is very similar to the 32x16 panel, with pin "D" replacing one ground connection.

  This is the layout we'll be referencing most often.

  If you have a 32x32 panel with *no pin labels at all,* then use this layout.

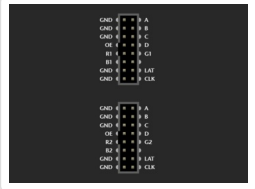
- "**Variant B**" for **32x32** and **64x32** panels. **The wiring is *identical* to Variant A above, *only the labels* are different.**

  Ground pins aren't labeled, but still need to be connected.

  LAT (latch) is labeled STB (strobe) here. R1/G1/B1/R2/G2/B2 are changed to R0/G0/B0/R1/G1/B1…but again, no functional difference, it's just ink.

  Our earliest **32x32** panels had a **two-socket** design, let's call it "**Variant C**." All the same pin functions are present but the layout is very different.

- 

  R/G/B on the **upper** socket correspond to R1/G1/B1 in Variant A. R/G/B on the **lower** socket correspond to R2/G2/B2.

  All the other signals (A/B/C/D/CLK/LAT/OE) need to be connected to **both**

sockets — e.g. one pin on the Arduino drives both CLK pins, and so forth.

Connecting to Arduino

There are two methods for connecting a matrix to an Arduino:

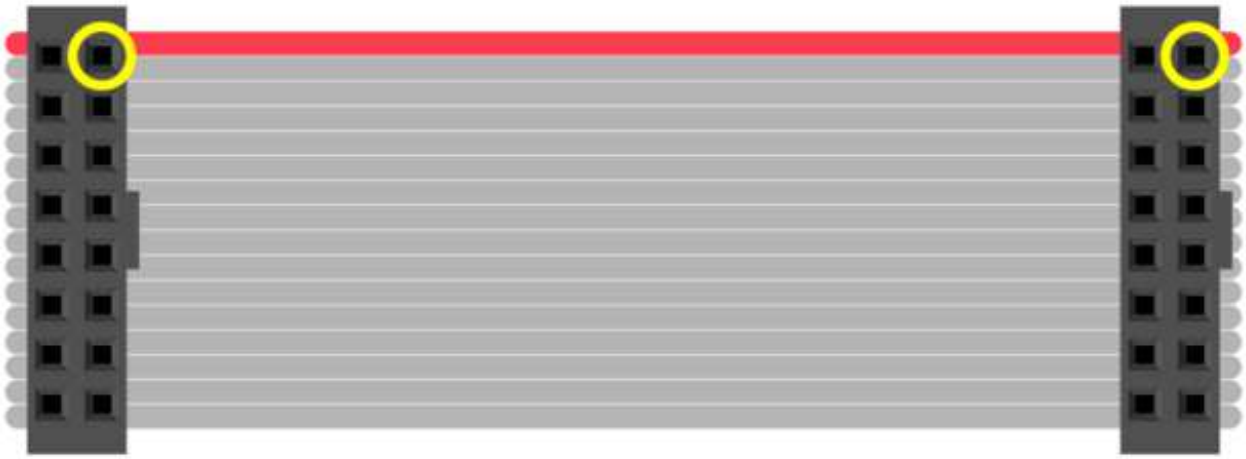
1. **Jumper wires** inserted between Arduino headers and a ribbon cable — this works well for testing and prototyping, but is not durable.
2. Building a ***proto shield*** — this is best for permanent installations.

These panels are normally run by very fast processors or FPGAs, not a 16 MHz Arduino. To achieve reasonable performance in this limited environment, our software is optimized by **tying specific signals to specific Arduino pins**. A *few* control lines can be reconfigured, but others are very specific…**you can't wire the whole thing willy-nilly.** The next two pages demonstrate compatible wiring…one using jumper wires, the other a proto shield…

Connecting with Jumper Wires

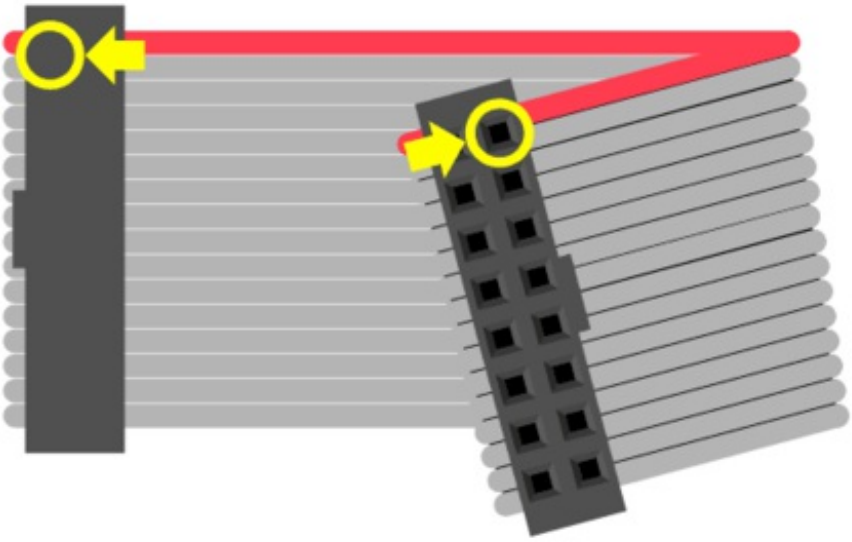
Ribbon cables and their corresponding headers are sometimes a topological puzzle. Here's a trick to help keep track…

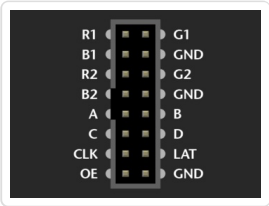
If you hold the ribbon cable flat — no folds — and with both connectors facing you, keys pointed the same direction — now there is a 1:1 correlation between the pins. The top-right pin on one plug links to the top-right on the other plug, and so forth. This holds true even if the cable has a doubled-over strain relief. **As long as the keys point the same way and the plugs face the same way, pins are in the same positions at both ends.**



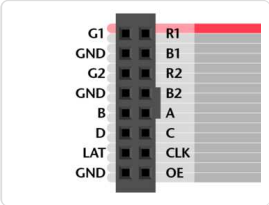
Plugged into a socket on the LED matrix, one header now faces *away* from you. If you double the cable back on itself (not a twist, but a fold)…to access a specific pin on the socket, the left and right columns are now mirrored (rows are in the same order — the red stripe provides a point of reference). You're looking "up" into the plug rather than "down" into the socket.

For example, R1 (the top-left pin on the INPUT socket) appears at the top-*right* of the exposed plug. You can jam a wire jumper in that hole to a corresponding pin on the Arduino…



- 

So! From the prior page, refer to the socket that's correct for your matrix type. The labels may be a little different (or none at all), but most are pretty close to what's shown here.

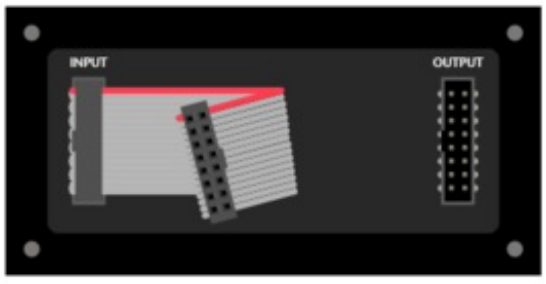
- 

Then *swap the columns* to find the correct position for a given signal.

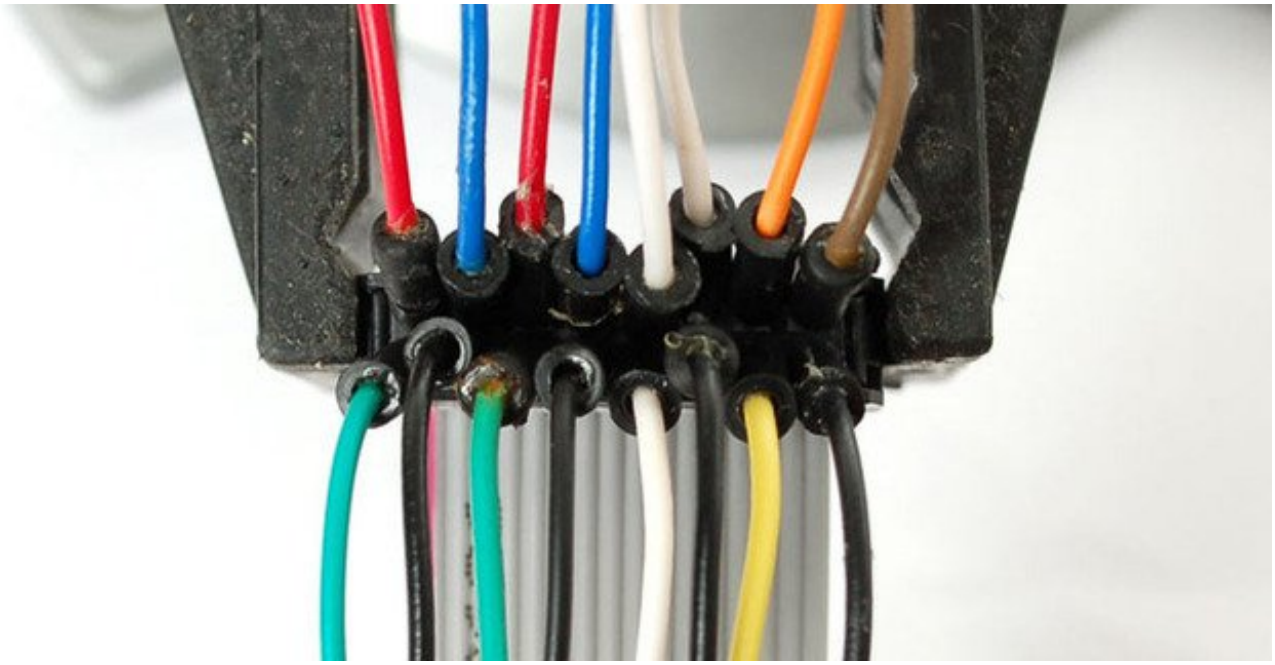
Either end of the ribbon cable can be plugged into the matrix INPUT socket. Notice below, the "key" faces the same way regardless.

With the free end of the ribbon toward the center of the matrix, the Arduino can be hidden behind it.

With the free end of the ribbon off the side, it's easier to see both the front of the matrix and the Arduino simultaneously, for making additional connections or for troubleshooting.

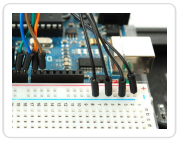


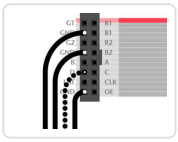
Using color-coded wires helps a *lot!* If you don't have colored wires, that's okay, just pay close attention where everything goes. Our goal is a fully-populated plug like this:

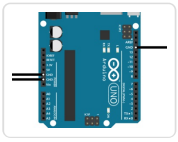


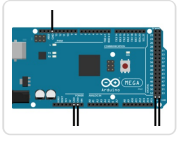
So! Let's proceed with the wiring, in groups…

Connect Ground Wires

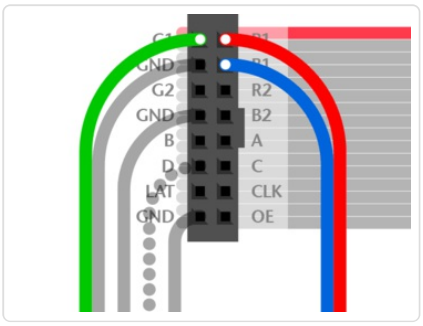
- 

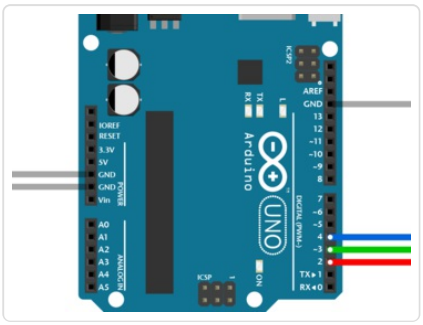
  32x32 and **64x32** matrices require **three** ground connections. **32x16** matrices have **four**.
- 

  Current **Arduino Uno** boards have **three** ground pins (the third is next to pin 13). If you need additional ground connections — for a 32x16 matrix, or if using an older Arduino board with only 2 ground pins — a solderless breadboard is handy for linking all these pins.
- 

  Arduino Mega boards have **five** ground pins. Same three as the Arduino Uno, plus two more next to pins 52 & 53.
- 

Upper RGB Data

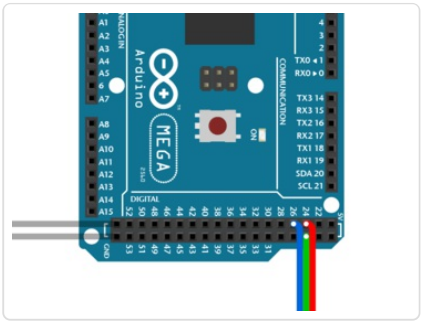
- 

- 

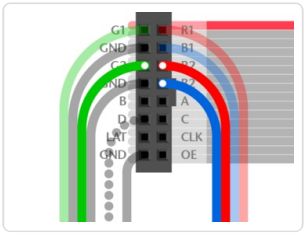
Pins **R1**, **G1** and **B1** (labeled R0, B0 and G0 on some matrices) deliver data to the **top half** of the display.

On the **Arduino Uno**, connect these to digital pins **2**, **3** and **4**.

On **Arduino Mega**, connect to pins **24**, **25** and **26**.

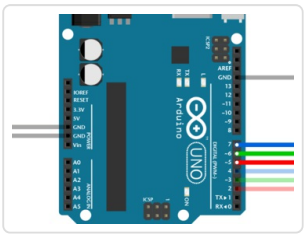
- 

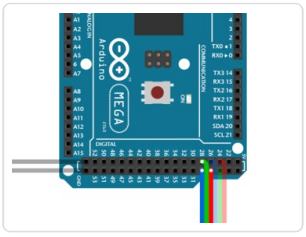
Lower RGB Data

- 

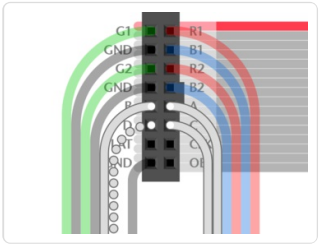
Pins **R2**, **G2** and **B2** (labeled R1, G1 and B1 on some matrices) deliver data to the **bottom half** of the display. These connect to the next three Arduino pins…

On **Arduino Uno**, that's pins **5**, **6** and **7**.

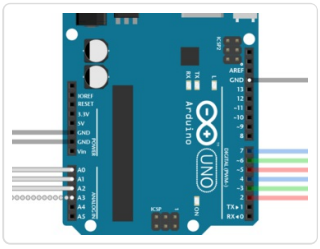
- 
On **Arduino Mega**, pins **27**, **28** and **29**.

- 

Row Select Lines

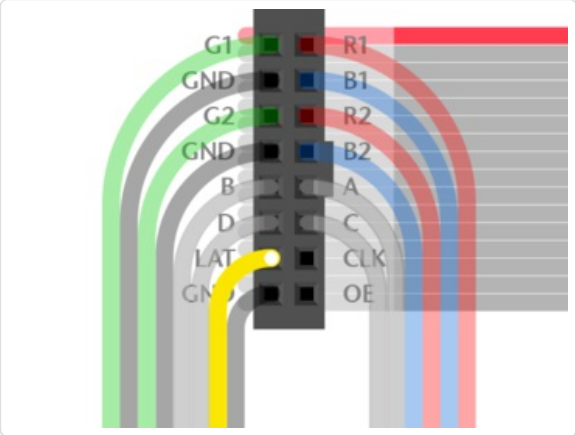
- 

Pins **A**, **B**, **C** and **D** select which two rows of the display are currently lit. (**32x16 matrices don't have a "D" pin** — it's connected to **ground** instead.)

- 
These connect to pins **A0**, **A1**, **A2** and (if D pin present) **A3**. This is the **same** for both the **Arduino Uno** and **Mega**.

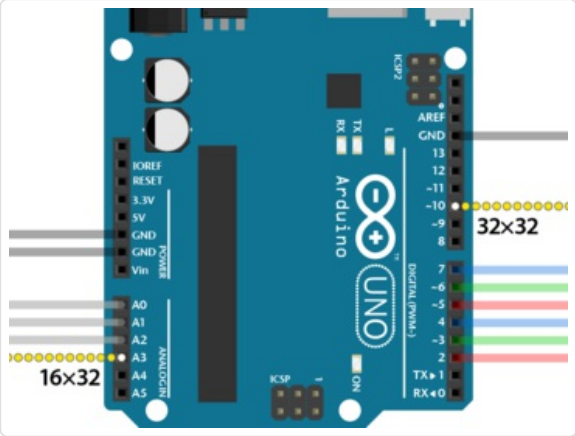
LAT Wire

LAT Wire

- 

For **32x32** and **64x32** matrices, **LAT** connects to Arduino pin **10**.

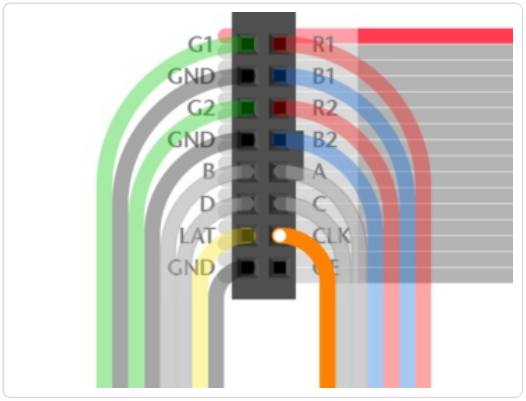
For a **32x16** matrix, use Arduino pin **A3**.

- 

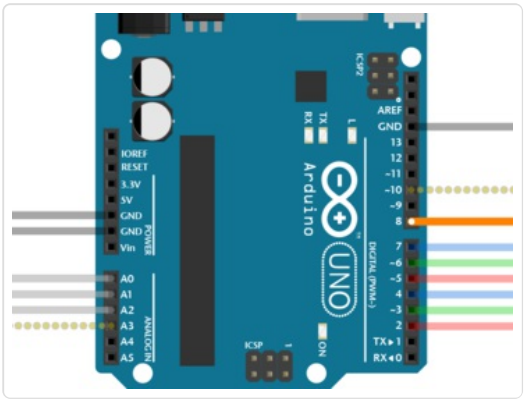
This is the same for **Arduino Uno** or **Mega**.

The LAT (latch) signal marks the end of a row of data.

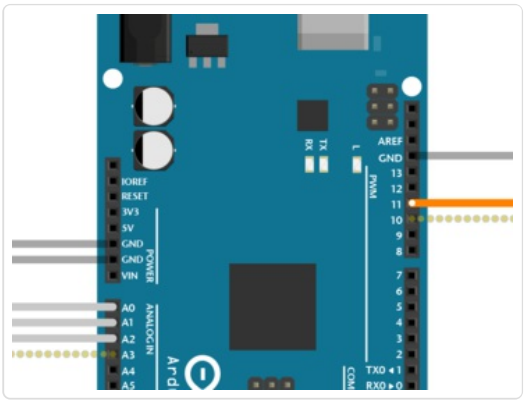
CLK Wire

- 

CLK connects to **pin 8** on an **Arduino Uno**, or **pin 11** on an **Arduino Mega**.

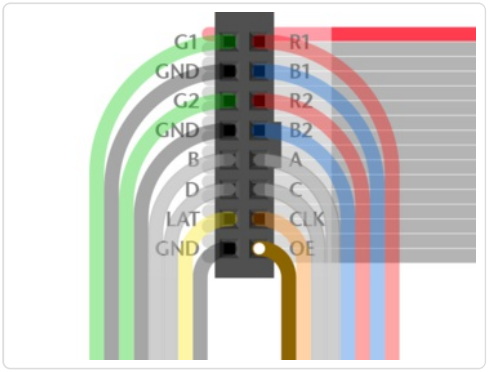
- 

The CLK (clock) signal marks the arrival of each bit of data.

- 

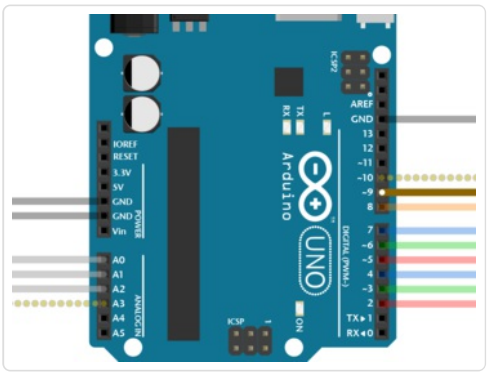
OE Wire

Last one!

- 

OE connects to Arduino pin **9**. This is the same for both the **Arduino Uno** and **Mega**.

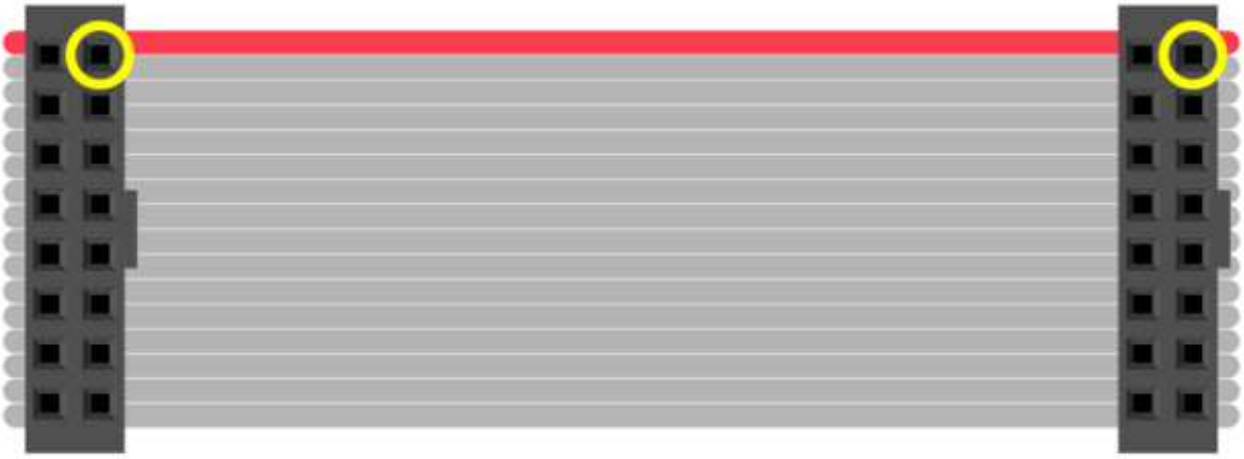
OE (output enable) switches the LEDs off when transitioning from one row to the next.

- 

That's it. You can skip ahead to the "Test Example Code" page now.

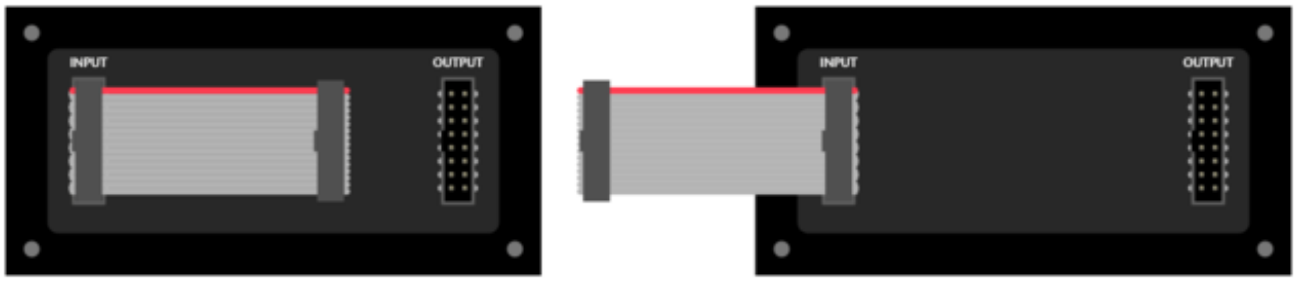
Connecting Using a Proto Shield

As mentioned on the “Jumper” page: if you hold a ribbon cable flat — no folds — and with both connectors facing you, keys pointed the same direction — there’s is a 1:1 correlation between the pins. The top-right pin on one plug links to the top-right on the other plug, and so forth. This holds true even if the cable has a doubled-over strain relief. **As long as the keys point the same way and the plugs face the same way, pins are in the same positions at both ends.**

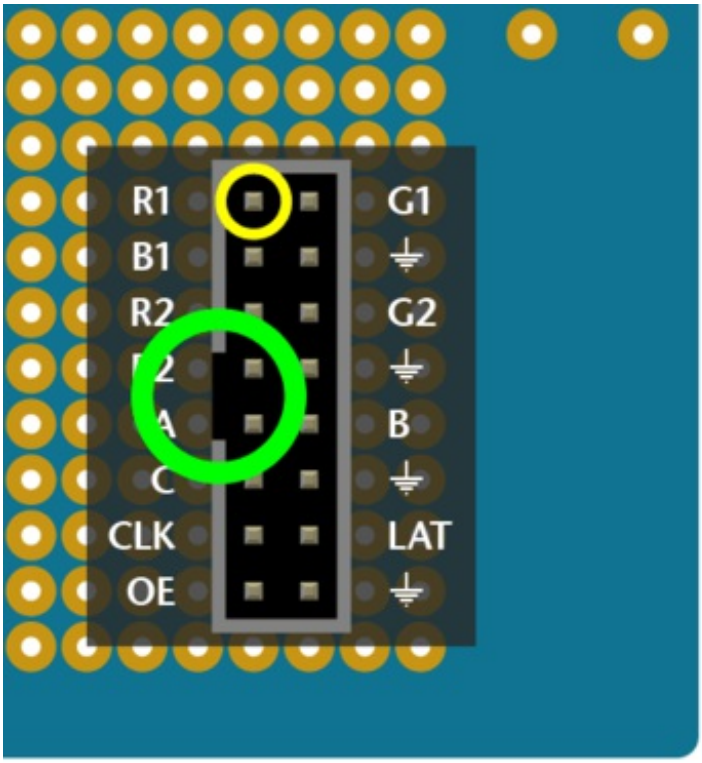


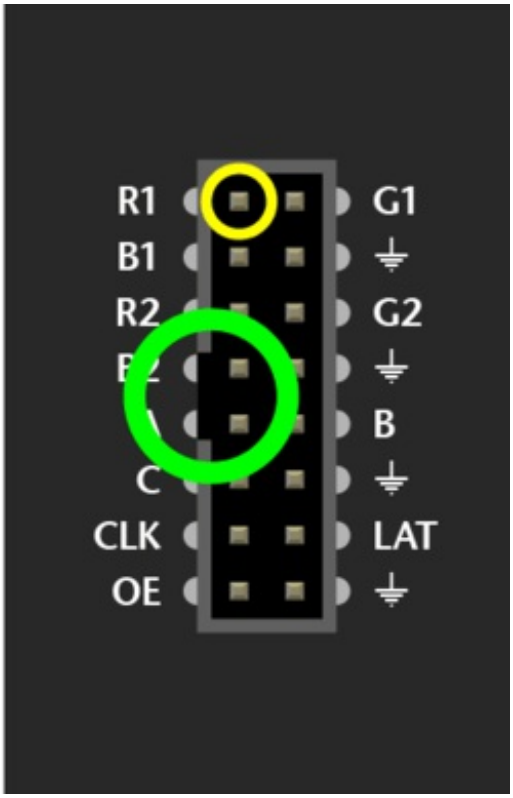
Either end of the ribbon cable can be plugged into the matrix INPUT socket.

The free end of the ribbon can point toward the center of the matrix, or hang off the side…the pinout is still the same. Notice below the direction of the “key” doesn’t change.



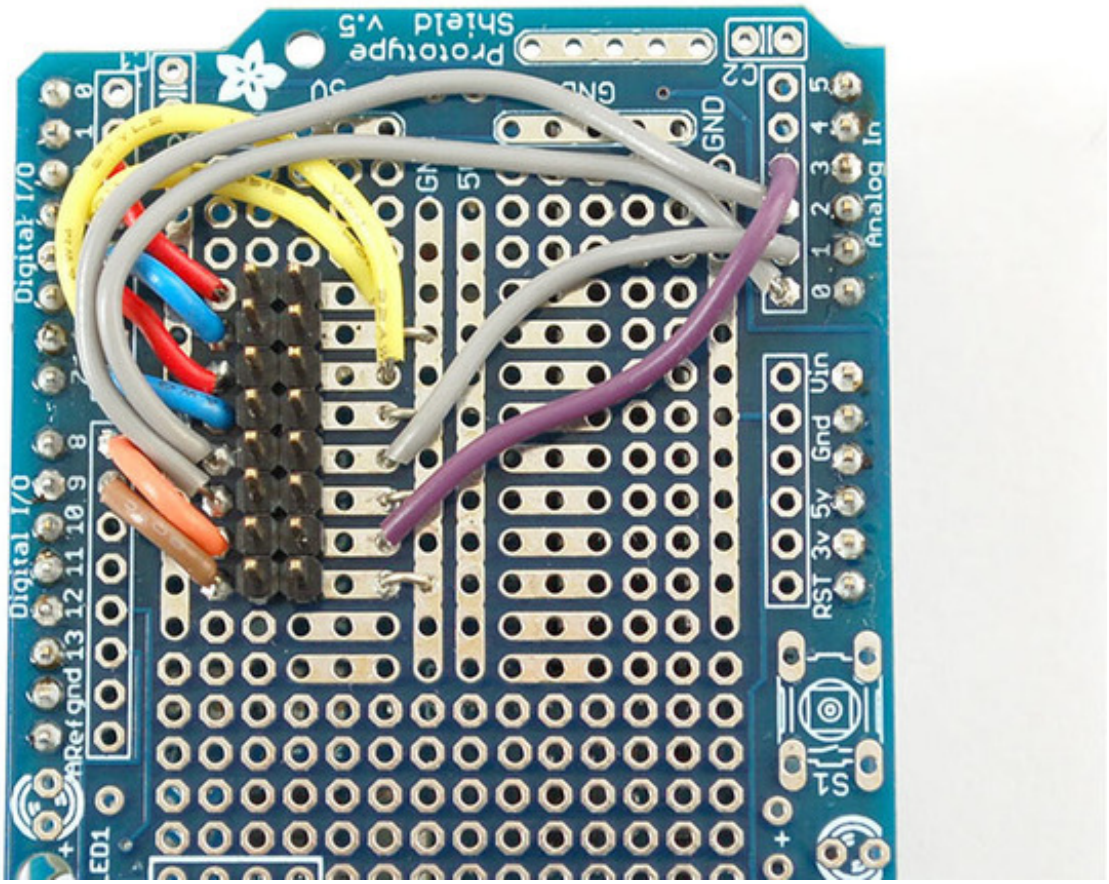
A dual-row header gets installed on the proto shield, similar to the connector on the matrix. Just like the ribbon cable lying flat, as long as these two headers are *aligned the same way,* they’ll **match pin-for-pin**; unlike the jumper wire method from the prior page, mirroring doesn’t happen.





Wires are then soldered from the header to specific Arduino pins on the proto shield. Try to keep wire lengths reasonably short to avoid signal interference.

Using color-coded wires helps a *lot!* If you don't have colored wires, that's okay, just pay close attention where everything goes. Our goal is a proto shield something like this:

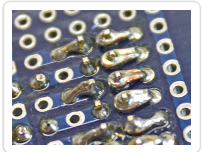


It's not necessary to install all the buttons and lights on the proto shield if you don't want — just the basic header pins are sufficient.

For **Arduino Uno**, using an **Adafruit** proto shield (http://adafru.it/eUM): **if using a shrouded socket (like on the back of the matrix — with the notch so a ribbon cable only fits one way) you'll need to place this near the "Reset" end of the shield.** The plastic shroud obscures a lot of pins. **Others' proto shields may be laid out different**…look around for a good location before committing to solder.

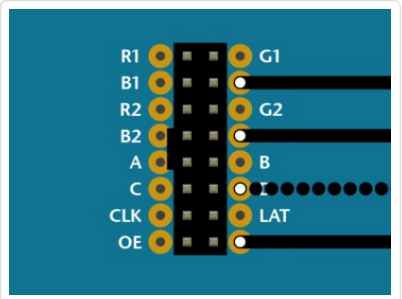
For **Arduino Mega** with our corresponding proto shield (http://adafru.it/192): a shrouded socket fits best near the **middle** of the shield.

Otherwise, you can use a plain 2x8-pin male header, or two 1x8 sections installed side-by-side (as in the photo above). Since there's no alignment key with this setup, you might want to indicate it with some tape or a permanent marker.

- 
Depending on the make and model of proto shield, some pins are designed to connect in short rows. Others don't. For the latter, strip a little extra insulation and bend the wire to wrap around the leg of the socket from behind, then solder.

Connect Ground Wires

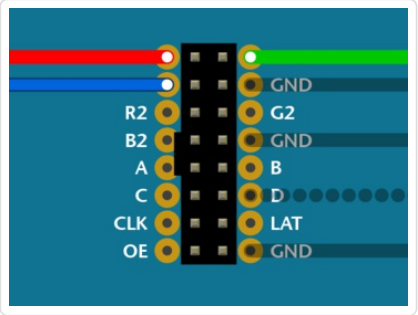
Connect Ground Wires

- 

32x32 and **64x32** matrices require **three** ground connections. **32x16** matrices have **four**.

Most proto shields have *tons* of grounding points, so you shouldn't have trouble finding places to connect these.

Upper RGB Data

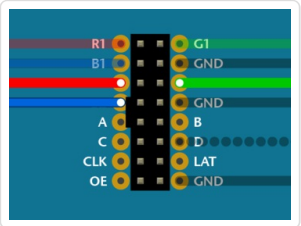
- 

Pins **R1**, **G1** and **B1** (labeled R0, B0 and G0 on some matrices) deliver data to the **top half** of the display.

On the **Arduino Uno**, connect these to digital pins **2**, **3** and **4**.

On **Arduino Mega**, connect to pins **24**, **25** and **26**.

Lower RGB Data

- 

Pins **R2**, **G2** and **B2** (labeled R1, G1 and B1 on some matrices) deliver data to the **bottom half** of the display. These connect to the next three Arduino pins…

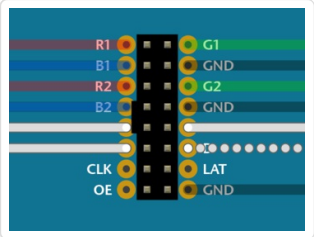
On **Arduino Uno**, that's pins **5**, **6** and **7**.

On **Arduino Mega**, pins **27**, **28** and **29**.

Row Select Lines

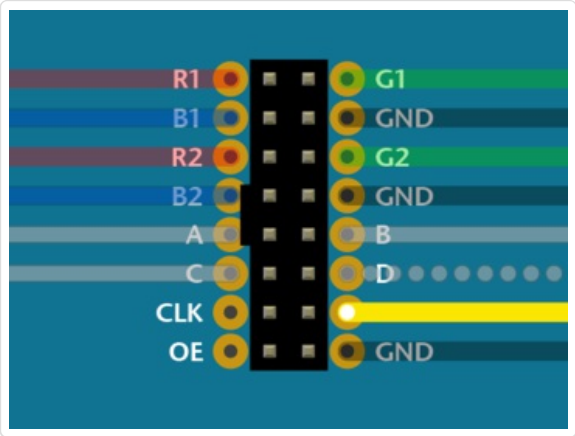
Pins **A**, **B**, **C** and **D** select which two rows of the display are currently lit. (**32x16 matrices don't have a "D" pin** — it's connected to

- **ground** instead.)



These connect to pins **A0**, **A1**, **A2** and (if D pin present) **A3**. This is the **same** for both the **Arduino Uno** and **Mega**.

LAT Wire

- 

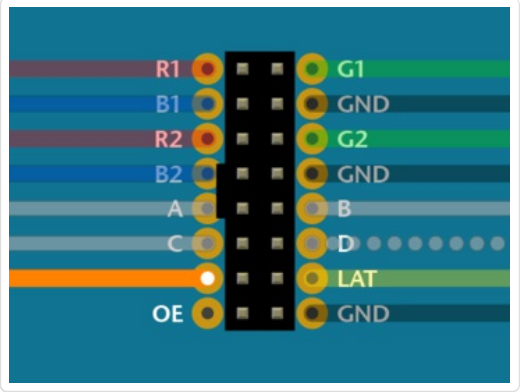
For **32x32** and **64x32** matrices, **LAT** connects to Arduino pin **10**.

For a **32x16** matrix, use Arduino pin **A3**.

This is the same for **Arduino Uno** or **Mega**.

The LAT (latch) signal marks the end of a row of data.

CLK Wire

- 

CLK connects to **pin 8** on an **Arduino Uno**, or **pin 11** on an **Arduino Mega**.

The CLK (clock) signal marks the arrival of each bit of data.

OE Wire

Last one!