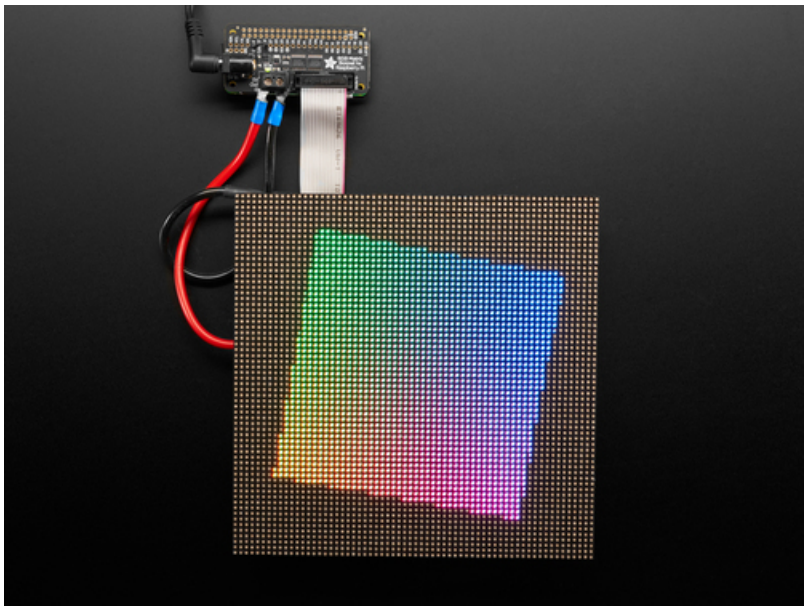
adafruit learning system

Adafruit RGB Matrix Bonnet for Raspberry Pi

Created by lady ada



Last updated on 2018-07-18 10:17:35 PM UTC

Guide Contents

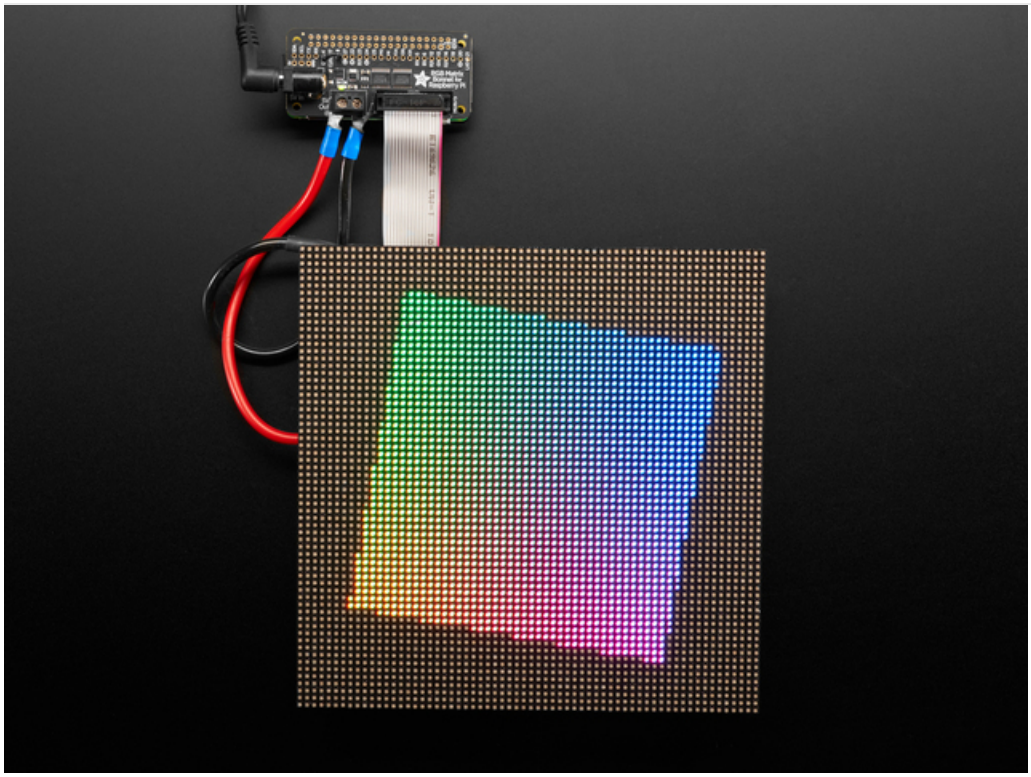
Guide Contents 2
Overview 3
Pinouts 7
5V protection circuitry and backpower diode 7
Matrix Drive pins 8
- Matrix Color Pins 9
- Matrix Control pins 9
- RGB Matrix Address pins 9

Driving Matrices 11
Step 1. Plug HAT/Bonnet into Raspberry Pi 11
Step 2. Connect Matrix Power cable to terminal block 12
Step 3. Connect RGB Matrix Data cable to IDC 14
Step 4. Power up your Pi via MicroUSB (optional but suggested) 15
Step 5. Plug in the 5V DC power for the Matrix 16
Check that the Matrix plugs are installed and in the right location 16
Step 6. Log into your Pi to install and run software 17
Testing the Examples 19
- The demos kinda run, but I'm seeing weird rectangles and glitches. 20

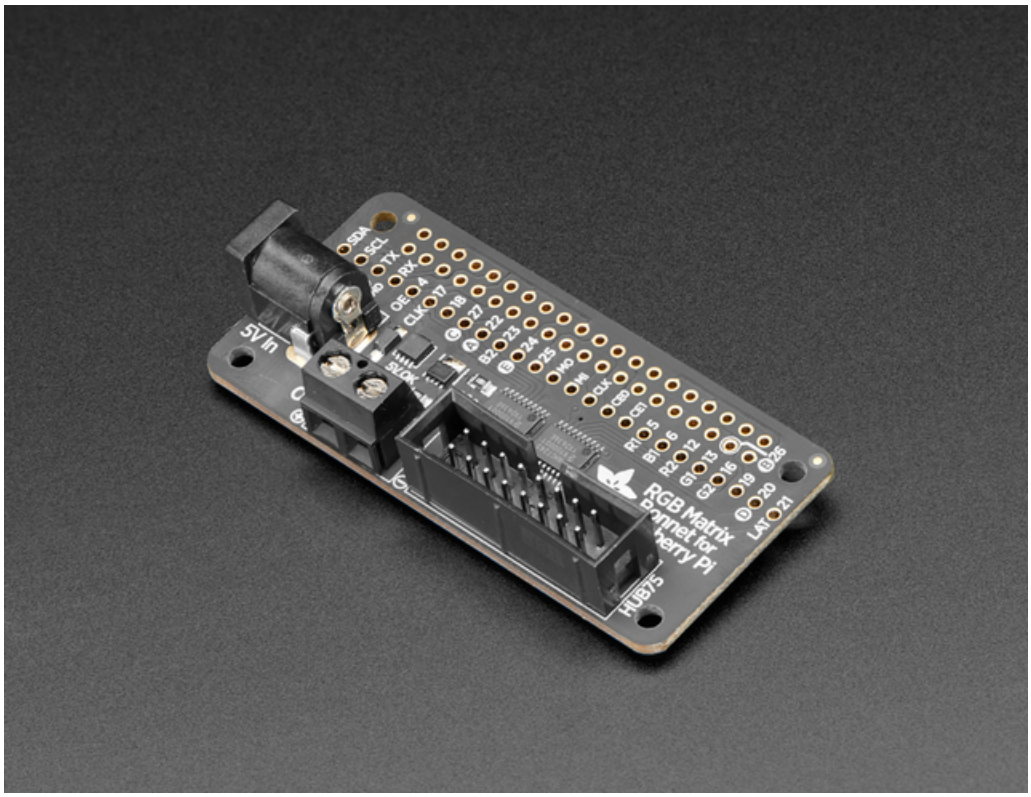
Using the Python Library 20
HELP! 22
- I'm using a Raspberry Pi 2 and things are all not working right! 22

Downloads 23
Datasheets 23
Schematic & Fabrication Print 23

Overview

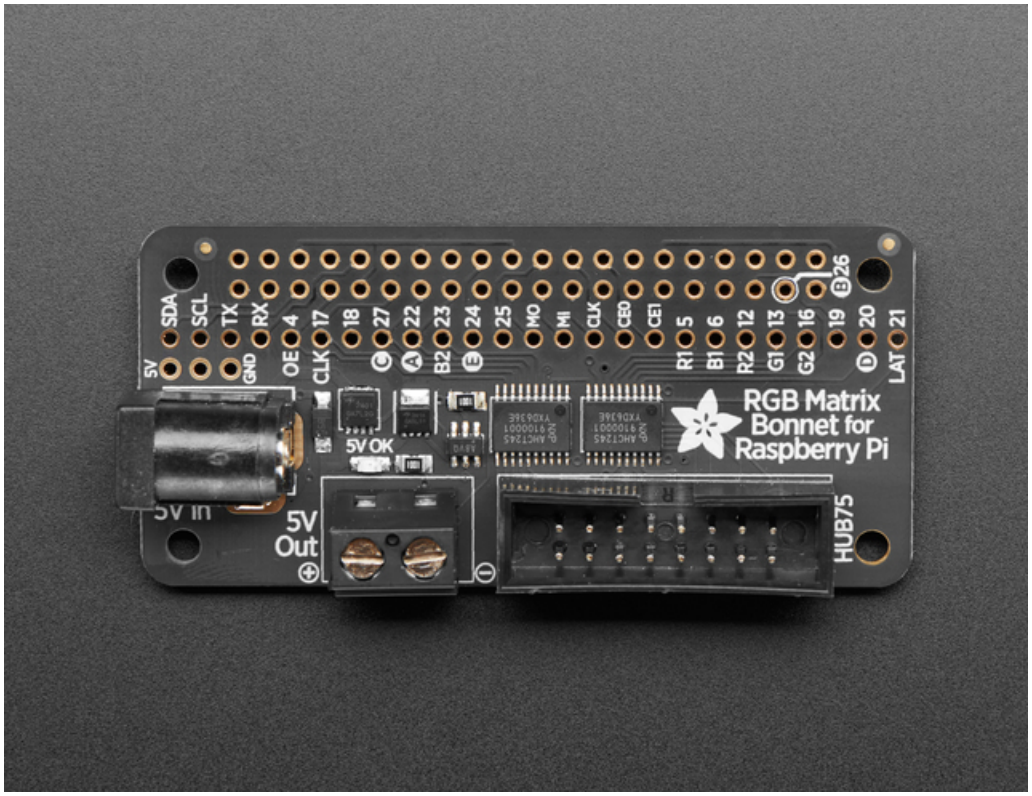


You can now create a dazzling display with your Raspberry Pi with the Adafruit RGB Matrix Bonnet. These boards plug into your Pi and makes it super easy to control RGB matrices such as those we stock in the shop and create a colorful scrolling display or mini LED wall with ease.



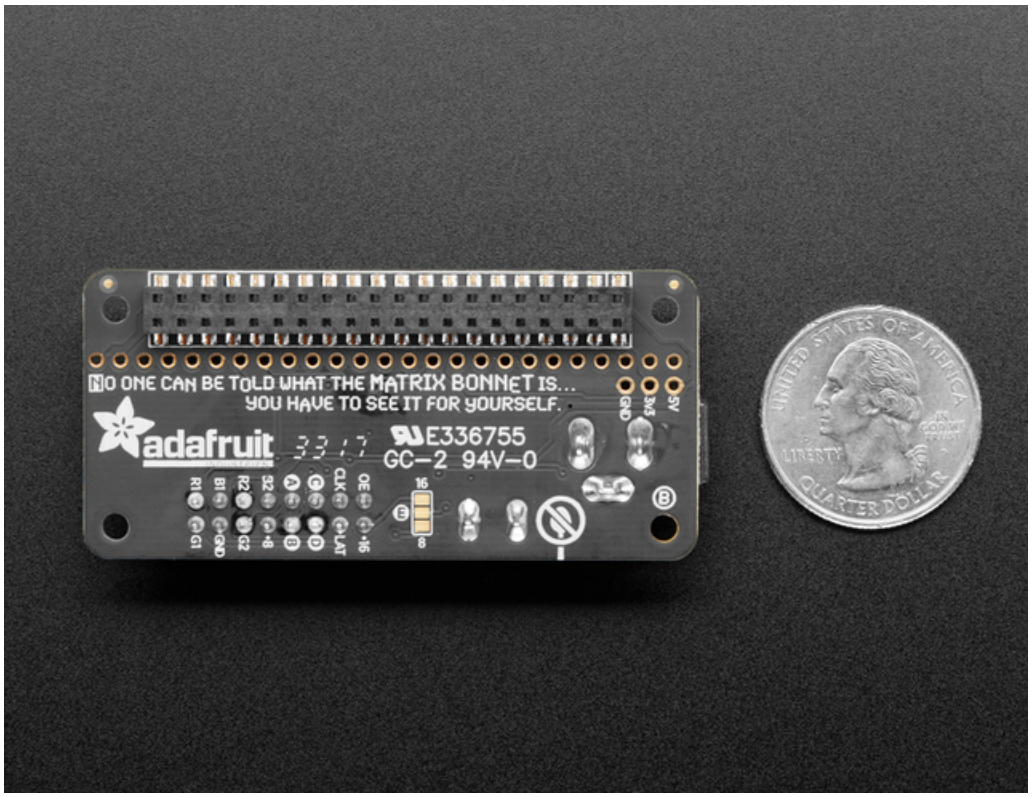
"Bonnet" boards work on any Raspberry Pi with a 40-pin GPIO header — Zero, Zero W/WH, Model A+, B+, Pi 2 and Pi 3. They *do not work* with older 26-pin boards like the original Model A or B. Note with the Pi Zero you may need to solder a header on the Pi board; it's normally unpopulated on that model.

We also have an older, HAT version of our RGB Matrix design. (https://adafru.it/eCC) The HAT version does not come fully assembled, does not support 1/32-scan matrices, but *does* come with a real time clock (RTC)



This bonnet will make your matrix projects super easy and avoids wiring complexity. Let me break it down for you:

- **Simple design** - plug in power, plug in IDC cable, run our Python code!
- **Power protection circuitry** - you can plug a 5V 4A wall adapter into the HAT and it will automatically protect against negative, over or under-voltages! Yay for no accidental destruction of your setup.
- **Onboard level shifters** to convert the RasPi's 3.3V to 5.0V logic for clean and glitch free matrix driving
- **Fully assembled compact design** no soldering required! Plugs onto any Raspberry Pi with a 2x20 connector, and you're ready to glow.



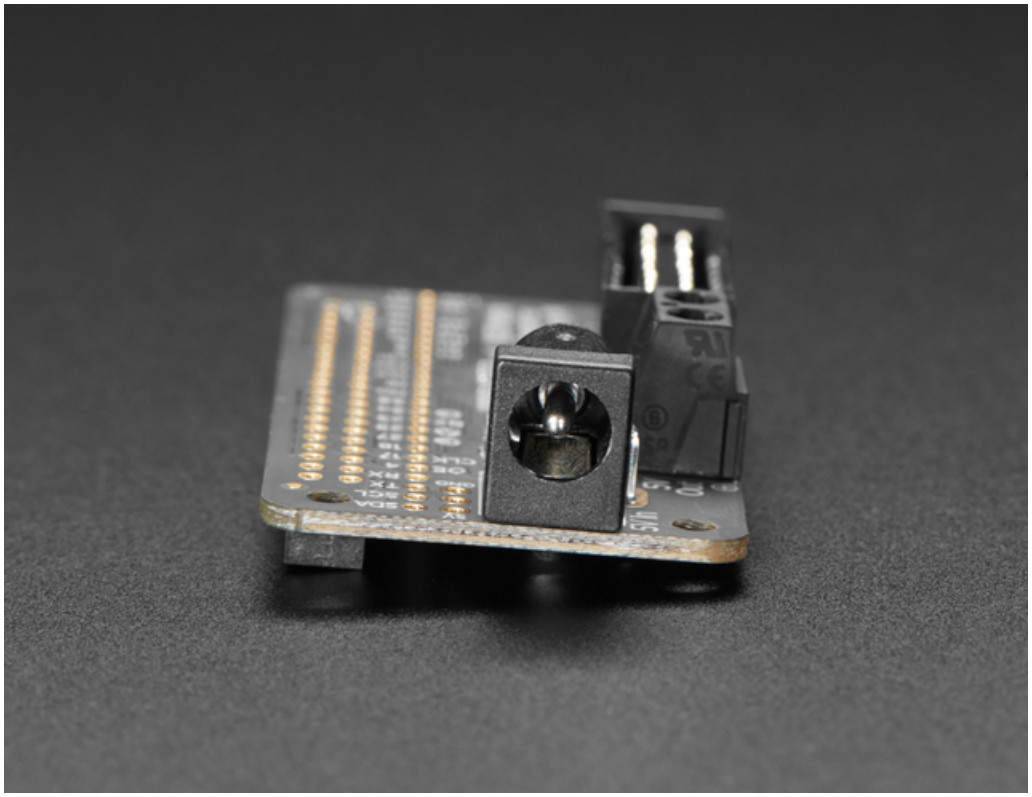
Works with any of our 16x32, 32x32 or 32x64 RGB LED Matrices with HUB75 connections(https://adafru.it/emd). You can also use 64x64 matrices by soldering a small jumper on the PCB (https://adafru.it/BhY). Want *more lights?* No problem, chain multiple matrices together for a longer display - we've only tested up to 32x128 but it works just fine. The bigger the display the harder it is on the Pi, so keep that in mind if you're using a lower-powered Pi Zero.

Please note: this Bonnet is only for use with HUB75 type RGB matrices. Not for use with NeoPixel, DotStar, or other 'addressable' LEDs.

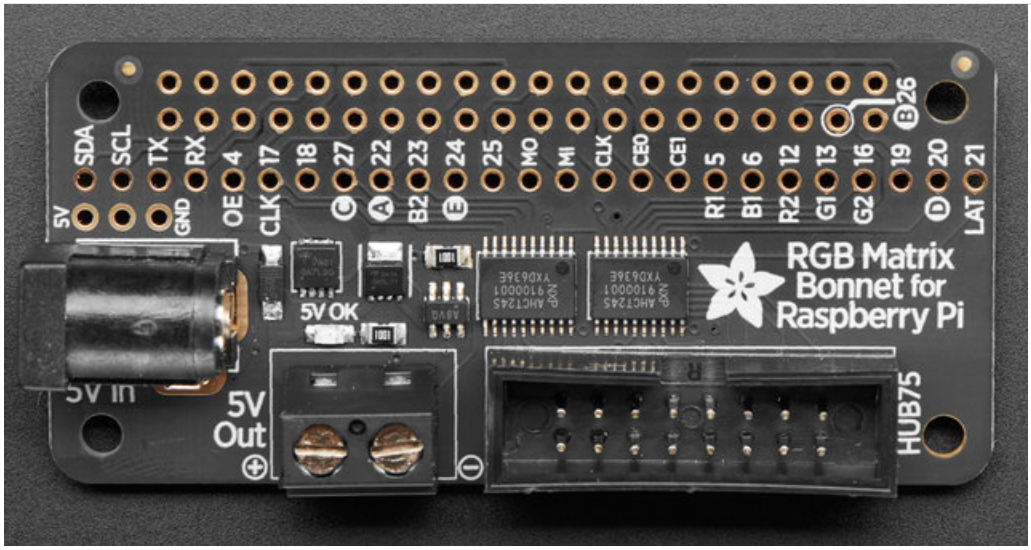
Each order comes with a fully assembled and ready to go bonnet with all parts assembled. **RGB Matrix is not included**, please check out our fine selection (https://adafru.it/emd)

A 5V power supply is also required, not included, for power the matrix itself, the Pi cannot do it, to calculate the power, multiply the width of all the chained matrices * 0.12 Amps : A 32 pixel wide matrix can end up drawing 32*0.12 = 3.85A so pick up a 5V 4A power supply (https://adafru.it/e50). We also stock 5V 10A power supplies (https://adafru.it/eCD). Actual power usage will vary with how many LEDs you light up at once.

Raspberry Pi not included (but we have 'em in the shop so pick one up, Zero, Model A+, B+, Pi 2 or Pi 3) (https://adafru.it/eme)



Pinouts



This Bonnet uses **a lot** of pins to drive the RGB Matrix. You'll still have a couple left over but just be aware a majority are in use by the matrix.

Unused GPIO pins include: SCL, SDA, RX, TX, #18, #25, MOSI, MISO, SCLK, CE0, CE1, #19.

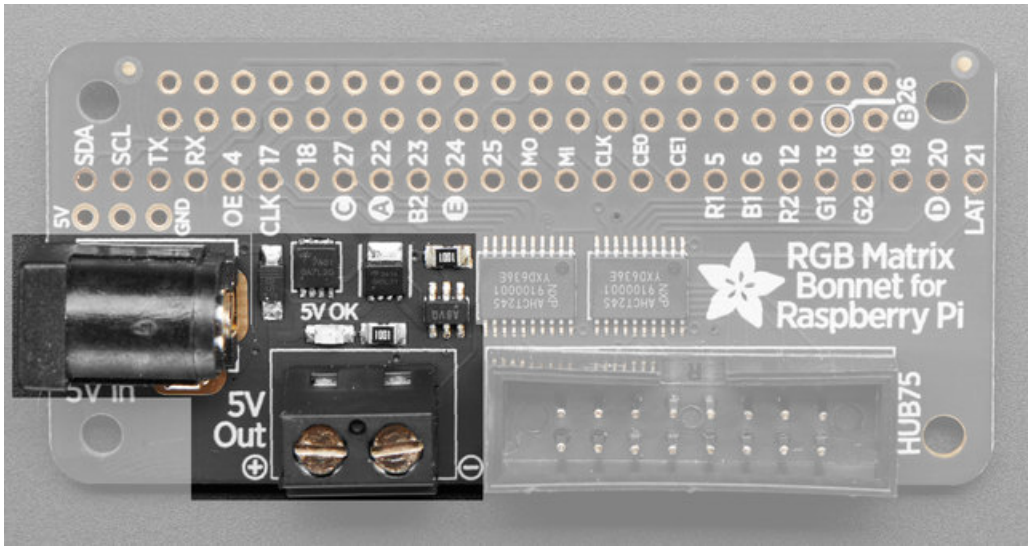
#24 is unused if you are not using a 1/32 scan matrix.

5V protection circuitry and backpower diode

LED matrix panels require 5V power and **a lot of it!** 5V 2A at a minimum and you can easily need a 5V 4A or 5V 10A supply for big stretches of panels!

Each matrix has 64 pixels (16x32 or 32x32 panels) or 128 pixels (for the 32x64 panels) lit at one time. Each pixel can draw up to 0.06 Amps each if on full white. The total max per panel is thus **64** * **0.06** = **3.95 Amps** or **128** * **0.06** = **7.68 Amps**

That's if all the LEDs are on at once, which is not likely - but still, its good to have at least half for the power supply in case you get bright!



5V power from a wall plug goes into the DC jack on the bonnet which then goes through a fancy protection circuit that makes sure the voltage is not higher than 5.8V - this means that if you accidentally grab a 9V or 12V plug or a reverse polarity plug you will not damage the bonnet, Pi and panels. (**Please note, this does not protect against extreme damage**, if you plug in a 120VAC output into the DC jack or continuously try to plug in the wrong voltage you could still cause damage so please do be careful!)

We recommend powering your driving Raspberry Pi from the Pi's microUSB port but we do have a 1A diode on board that will automatically power the Pi if/when the voltage drops. So if you want, just plug in the 5V wall adapter into the bonnet and it will automagically power up the Pi too!

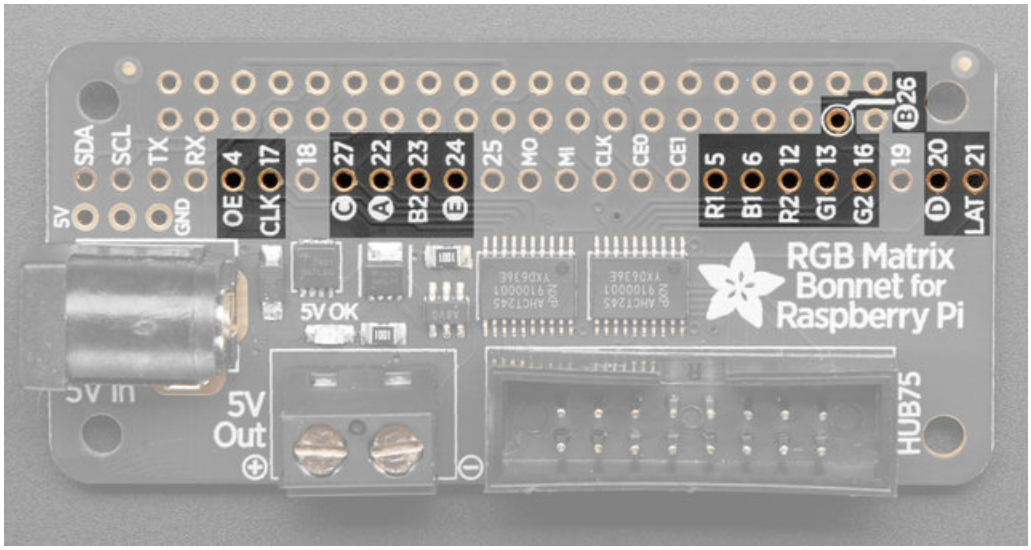
The green LED next to the DC jack will indicate that the 5V power is good, make sure it is lit when trying to use the bonnet!

Matrix Drive pins

The matrix does not work like 'smart' pixels you may have used, like NeoPixels or DotStars or LPD8806 or WS2801 or what have you. The matrix panels are very 'dumb' and have no memory or self-drawing capability.

Data must be constantly streamed to the matrix for an image to display! So all of these pins are always used when drawing to the display

All these pins go thru a 74AHCT245 level shifter to convert the 3.3V logic from the Pi to the 5V logic required by the panels



Matrix Color Pins

- Pi GPIO **#5 - Matrix R1** (Red row 1) pin
  This pin controls the red LEDs on the top half of the display
- Pi GPIO **#13 - Matrix G1** (Green row 1) pin
  This pin controls the green LEDs on the top half of the display
- Pi GPIO **#6 - Matrix B1** (Blue row 1) pin
  This pin controls the blue LEDs on the top half of the display
- Pi GPIO **#12 - Matrix R2** (Red row 2) pin
  This pin controls the red LEDs on the bottom half of the display
- Pi GPIO **#16 - Matrix G2** (Green row2) pin
  This pin controls the green LEDs on the bottom half of the display
- Pi GPIO **#23 - Matrix B2** (Blue row 2) pin
  This pin controls the blue LEDs on the bottom half of the display

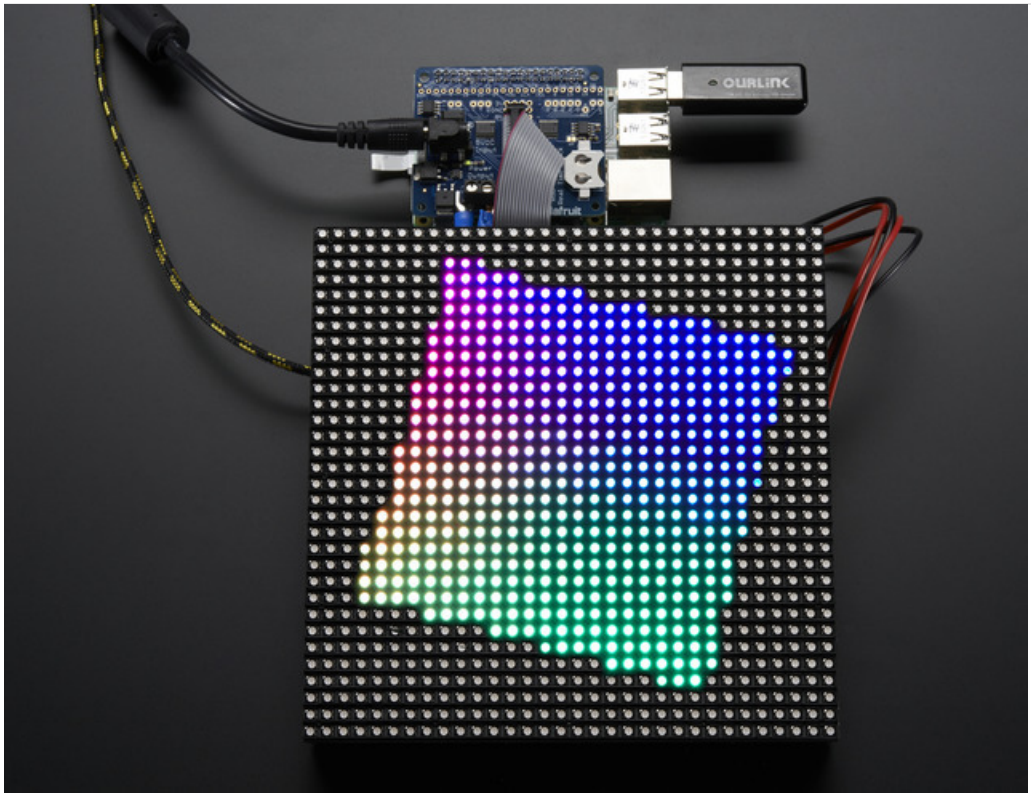
Matrix Control pins

- Pi GPIO **#4 - Matrix OE** (output enable) pin
  This pin controls whether the LEDs are lit at all
- Pi GPIO **#17 - Matrix CLK** (clock) pin
  This pin is the high speed clock pin for clocking RGB data to the matrix
- Pi GPIO **#21 - Matrix LAT** (latch) pin
  This pin is the data latching pin for clocking RGB data to the matrix

RGB Matrix Address pins

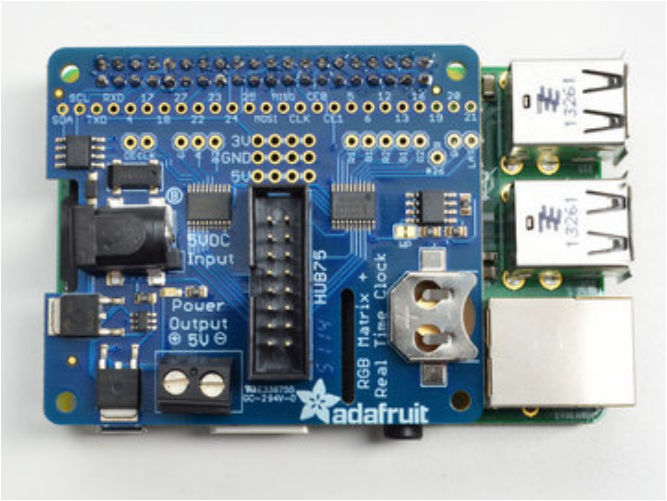
- Pi GPIO **#22 - Matrix A** (address A) pin
  This pin is part of the 1->32, 1->16 or 1->8 multiplexing circuitry.
- Pi GPIO **#26 - Matrix B** (address B) pin
  This pin is part of the 1->32, 1->16 or 1->8 multiplexing circuitry.
- Pi GPIO **#27 - Matrix C** (address C) pin
  This pin is part of the 1->32, 1->16 or 1->8 multiplexing circuitry.
- Pi GPIO **#20 - Matrix D** (address D) pin
  This pin is part of the 1->32, 1->16 multiplexing circuitry. Used for 32-pixel and 64-pixel tall displays only
- Pi GPIO **#24 - Matrix E** (address E) pin
  This pin is part of the 1->32 multiplexing circuitry. Used for 64-pixel tall displays only

Driving Matrices

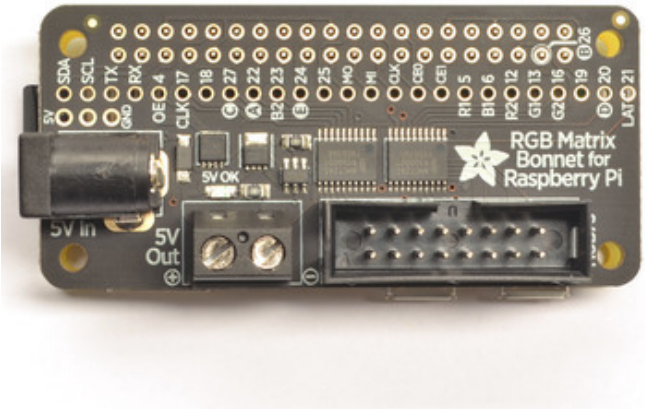


OK we're onto the fun part now! Be sure you have completed the Assembly step before continuing, the soldering is **not optional**

Step 1. Plug HAT/Bonnet into Raspberry Pi

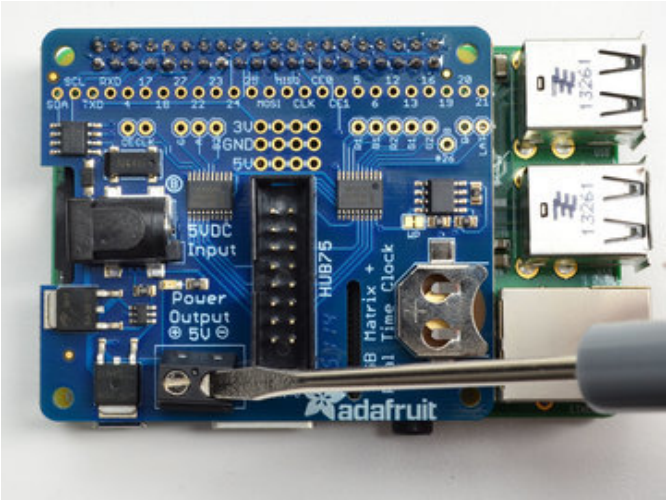


Shut down your Pi and remove power. Plug the HAT or Bonnet on so all the 2x20 pins go into the GPIO header.

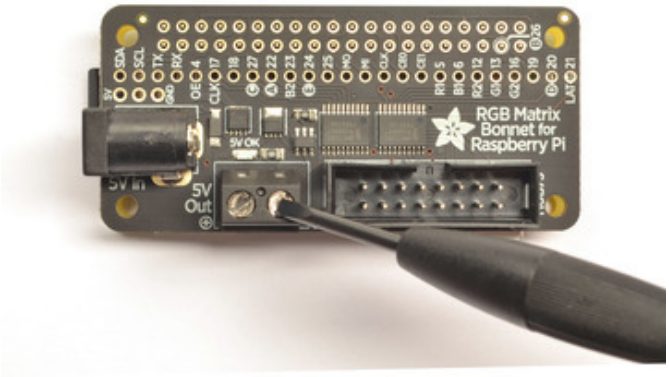


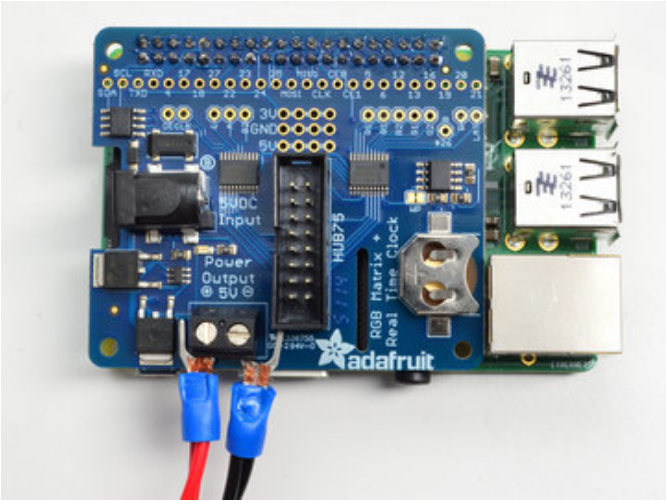
Step 2. Connect Matrix Power cable to terminal block

Your RGB matrix came with a red & black power cable. One end has a 4-pin MOLEX connector that goes into the matrix. The other end probably has a spade connector. If you didn't get a spade connector, you may have to cut off the connector and tin the wires to plug them into the terminal block

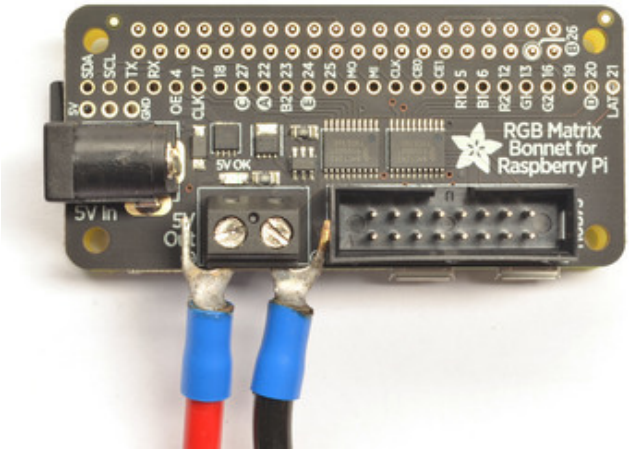


Either way, unscrew the terminal blocks to loosen them

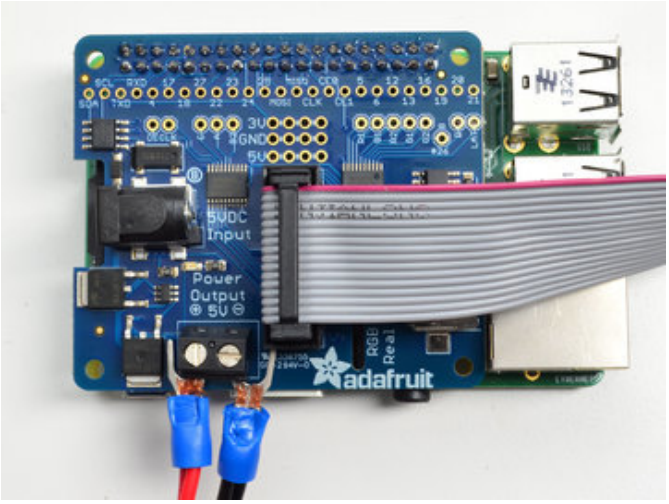




Plug the red wire into the + side, and the black wire into the - side.

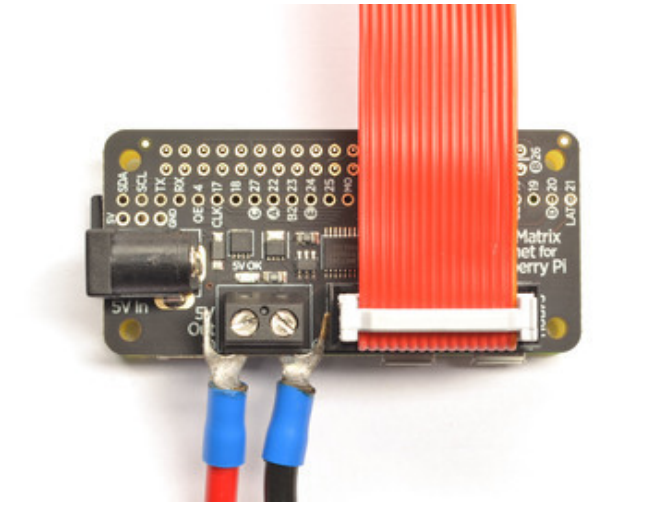


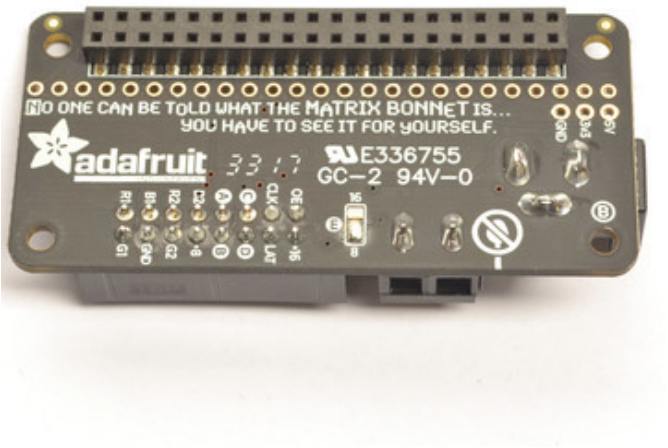
Step 3. Connect RGB Matrix Data cable to IDC



The RGB matrix also came with a 2x8 data cable. Connect one end to the matrix's INPUT side and the other end to the IDC socket on the HAT/bonnet.

It wont damage the matrix if you accidentally get the cable connected to the output end of the matrix but it wont work so you might as well get it right first time!





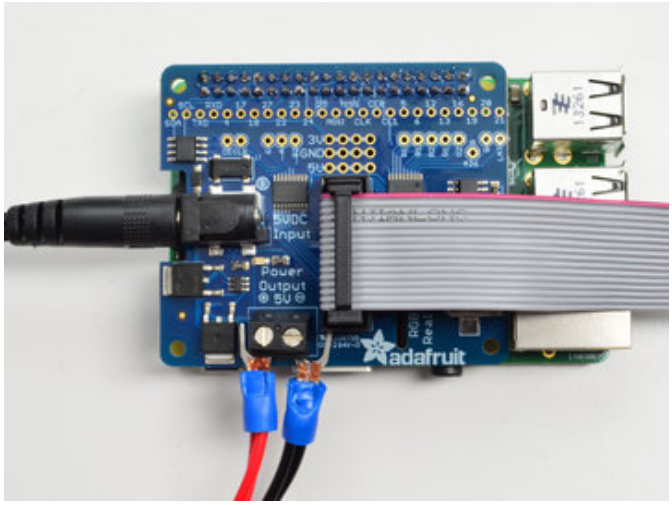
If you're using a 64x64 RGB matrix and a bonnet, use your soldering iron to melt a blob of solder on the bottom solder jumper so the middle pad is 'shorted' to **8**

Step 4. Power up your Pi via MicroUSB (optional but suggested)

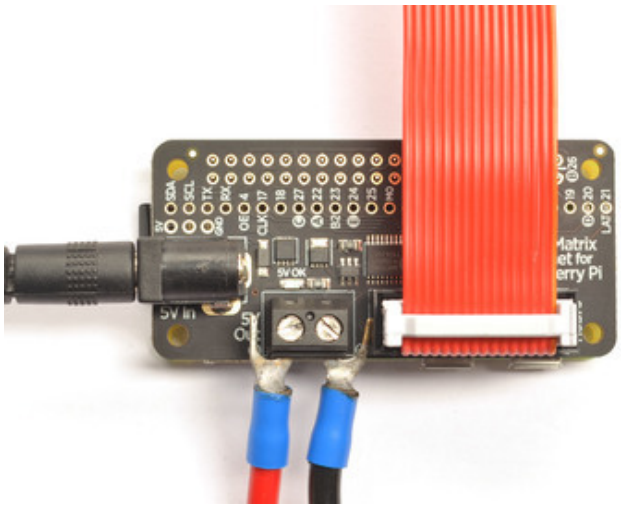
Connect your Raspberry Pi to power via the microUSB cable, just like you normally would to power it up.

You **can** power the Pi via the 5V wall plug that is also used for the Matrix but its best to have it powered seperately

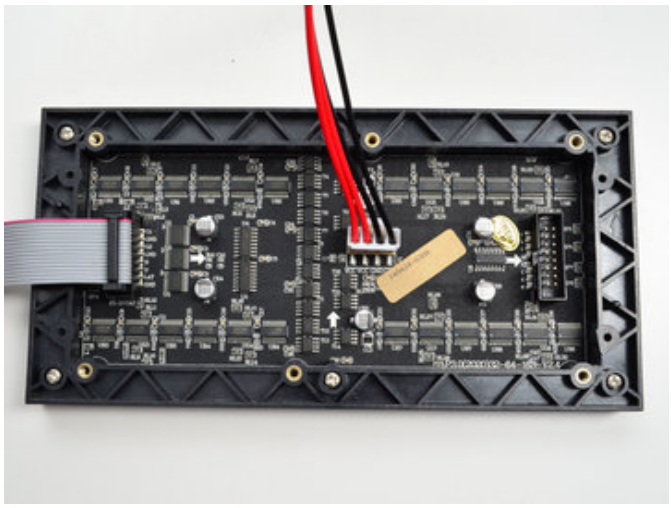
Step 5. Plug in the 5V DC power for the Matrix



OK now you can plug in your 5V 2A or 4A or larger wall adapter into the HAT/bonnet. This will turn the green LED on but nothing will display on your matrix yet because no software is running!



Check that the Matrix plugs are installed and in the right location



IDC goes into the INPUT side (look for any arrows, arrows point from INPUT side to OUTPUT)

Power plug installed, red wires go to VCC, black wires to GND

Step 6. Log into your Pi to install and run software

OK now you are ready to run the Pi software. You will need to get into a command line via the HDMI monitor, ssh or console cable. You will also need to make sure your Pi is on the Internet via a WiFi or Ethernet connection.

We have a script that downloads the code and any prerequisite software. It works with the current **Raspbian "Stretch" operating system** (either the Lite or Desktop version):

```
curl https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-Scripts/master/rgb-matrix.sh >rgb-
sudo bash rgb-matrix.sh
```

The LED-matrix library is (c) Henner Zeller h.zeller@acm.org with GNU General Public License Version 2.0 http://www.gnu.org/licenses/gpl-2.0.txt (https://adafru.it/ewN)

Earlier versions of this guide used our own fork of this library. That's **deprecated** now, but still available (https://adafru.it/ewy)if you have existing code built atop it. Otherwise, use this installer script and latest code.

```
pi@raspberrypi:~ $ curl -O https://raw.githubusercontent.com/adafruit/Raspberry-
Pi-Installer-Scripts/master/rgb-matrix.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  6025  100  6025    0     0  15257      0 --:--:-- --:--:-- --:--:-- 15291
pi@raspberrypi:~ $ sudo bash rgb-matrix.sh
```

When first run, the script will explain its plans and give you the option to cancel.

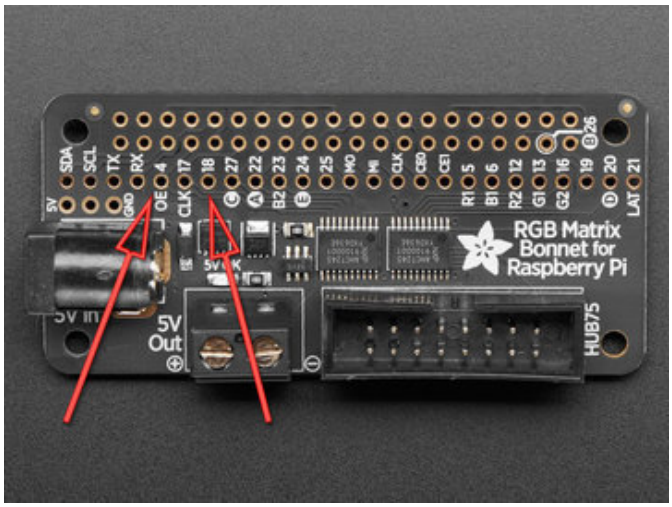
Of particular note here: any existing installation will be replaced. If there is a directory called “rpi-rgb-led-matrix” in the current working directory, its contents will be overwritten. Additionally, a Python module is installed and will replace anything currently there. If this is a problem, cancel and make a backup. Otherwise, sometimes reinstalling is exactly what you want.

```
This script installs software for the Adafruit
RGB Matrix Bonnet or HAT for Raspberry Pi.
Steps include:
- Update package index files (apt-get update)
- Install prerequisite software
- Install RGB matrix driver software
- Configure boot options
Run time ~10 minutes. Some options require reboot.
EXISTING INSTALLATION, IF ANY, WILL BE OVERWRITTEN.

CONTINUE? [y/N]
```

Next the script will ask you what kind of adapter you’re using between the Pi and RGB matrix: either an **Adafruit RGB Matrix Bonnet**, or **RGB Matrix HAT with RTC**. If you select the latter, you’ll also be asked if you want to install additional drivers for the realtime clock.

```
Select interface board type:
1. Adafruit RGB Matrix Bonnet
2. Adafruit RGB Matrix HAT + RTC

SELECT 1-2:
```

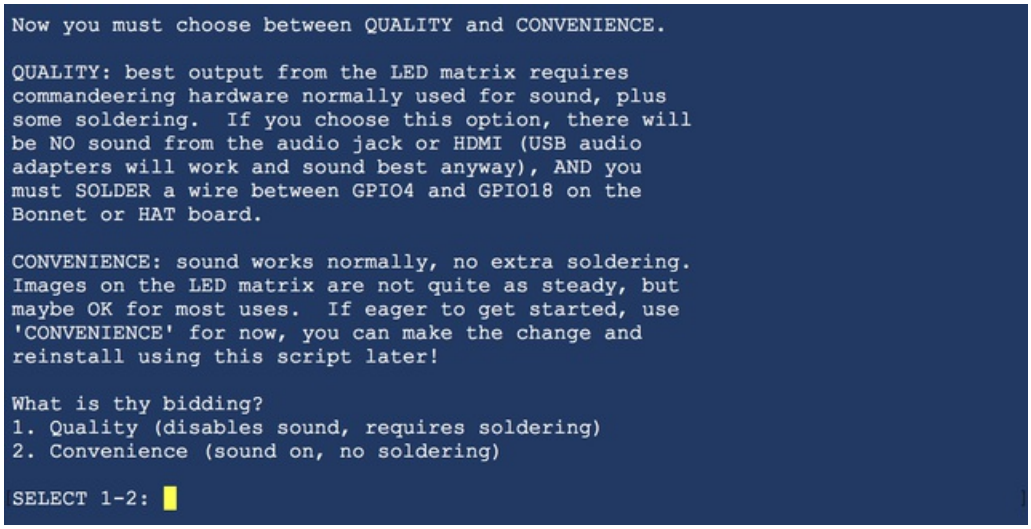
Then you’re asked whether you need the *absolute best image possible* from the LED matrix, or can accept slightly reduced quality for the sake of simplicity.



The “quality” option comes at a cost. First, you need to **solder a jumper wire** between **GPIO4** and **GPIO18** on the Bonnet or Hat board. Also, the normal sound output of the Raspberry Pi must be disabled. You can still use a USB sound adapter if needed, but **audio over HDMI or from the 1/8" jack will not be present.**

The “convenience” setting requires no changes and sound still works. For many casual projects this might look good enough. There’s a bit of flicker from the matrix, that’s all.

If you’re not sure, or if you just want to get started experimenting with your new gadget, select “convenience” for now. You can make the change and reinstall the software later if needed.



The script will **confirm your selections** and offer **one more chance to cancel** without changes.

There's a lot of software to update, download and install, so it may take up to**15 minutes** or so to complete. Afterward, you'll be asked whether you want to **reboot** the system. If you've selected to install RTC support (for the Matrix HAT + RTC) or have made a change in the "quality" vs "convenience" setting, a reboot is required.

All other settings (LED matrix size, number of "chained" matrices and so forth) are specified at run-time.

> Overclocked Raspberry Pi boards may produce visual glitches on the LED matrix. If you encounter such trouble, first thing to try is to set the Pi to the default (non-overclocked) speed using raspi-config, then reboot and retest.

Testing the Examples

The installer creates a directory called `rpi-rgb-led-matrix`, and inside this is a subdirectory `examples-api-use` with a few programs we can use to experiment with the matrix and confirm everything's working.

All of the examples — and any code using the companion libraries — accept a common set of command-line switches for specifying the LED matrix size and other options. Among the more vital options are:

`--led-rows=` (rows)

Specifies the number of rows (or height or the number of pixels vertically) of your LED matrix (or matrices, if you have several chained...they all need to be the same size though). Default value if unspecified is **32**. Maximum value with the Adafruit **RGB Matrix HAT + RTC** is **32**. Maximum with the **RGB Matrix Bonnet** is **64**.

`--led-cols=` (columns)

Specifies the number of columns (or width or the number of pixels horizontally) of your LED matrix/matrices. Default value if unspecified is **32**.

`--led-chain=` (chained)

Specifies the number of matrices in the chain...the output of one connects to the input of the next. Default value if unspecified is **1**.

Here's how to run one of the examples — a rotating colored square. Because this code is performing low-level hardware operations, it must be run using the `sudo` command:

```
sudo ./demo -D0 --led-rows=32 --led-cols=16
```

That's for a single 32x16 pixel RGB matrix. If you have a different size, change the `--led-rows` and/or `--led-cols` values. Add a `--led-chain` value if multiple matrices are chained.

There are 12 different examples in the demo program (0 through 11), chosen with `-D`. For a full list of the program's options, just type `demo`.

Depending on your matrix type and Raspberry Pi model, some additional options may need fine-tuning:

`--led-slowdown-gpio=` (0...2) Sometimes needed to "throttle back" the speed when using a fast Pi with a slower panel. Default is **1**.

`--led-rgb-sequence=` (RGB order) Some LED matrices may have their red, green and blue LEDs wired up in a different order...for example, if you need to swap the green and blue channels, use `--led-rgb-sequence=RBG`. Default is `RGB`.

`--led-pwm-bits=` (1...11) For long matrix chains you'll probably need to use fewer PWM bits, sacrificing some color fidelity to improve refresh speed. Default is `11`.

There are still many **additional options** but they're **increasingly esoteric** and might only be needed with RGB matrices from other sources. For a complete explanation of these options (and a more in-depth explanation of the options above) see the **documentation accompanying hzeller's code repository (https://adafru.it/kdg).**

The demos kinda run, but I'm seeing weird rectangles and glitches.

If your Pi is overclocked, or if you're using a Raspberry Pi 2, you may need to dial back the matrix control speed slightly. This can be done with the `--led-slowdown-gpio=2` setting. Conversely, early Raspberry Pis (Model A, B and similar) might get an improved image by speeding up the matrix code with a value of `0` here.

There are a few additional examples in that directory showing how to write C++ programs to draw to the matrix. Look through the source code and Makefile to see how this is done and how to link with the `rgbmatrix` library. And there's more documentation in the hzeller repository (https://adafru.it/BhV), including initializing the matrix size and chain length in your code so it's not necessary to specify this on the command line every time.

Using the Python Library

Some Python examples are included in the `rpi-rgb-led-matrix/bindings/python/samples` directory. The matrix installer script has already loaded the prerequisite Python Imaging Library and installed the `rgbmatrix` module for both Python 2.7 and Python 3.

Again, **more documentation is available in the library author's repository (https://adafru.it/BhW)**, and some of the examples show how to specify the matrix size and chain length in code rather than command-line selections every time.

While the `rgbmatrix` module provides its own drawing operations, it can also work with the Python Imaging Library as an "offscreen canvas" that's then issued to the matrix with the SetImage() or SwapOnVSync() function — see the examples with "image-" in their name.

Core PIL image functions are explained here: The Image Module (https://adafru.it/dvE)

Graphics functions (lines, etc.) are here: The ImageDraw Module (https://adafru.it/dfH)

Reminder: the older Adafruit fork of the RGB matrix library (https://adafru.it/ewy) is still available if you need it for existing code, but consider this deprecated. For new projects we recommend the more up-to-date hzeller code installed by the `rgb-matrix.sh` script!

HELP!

I'm using a Raspberry Pi 2 and things are all not working right!

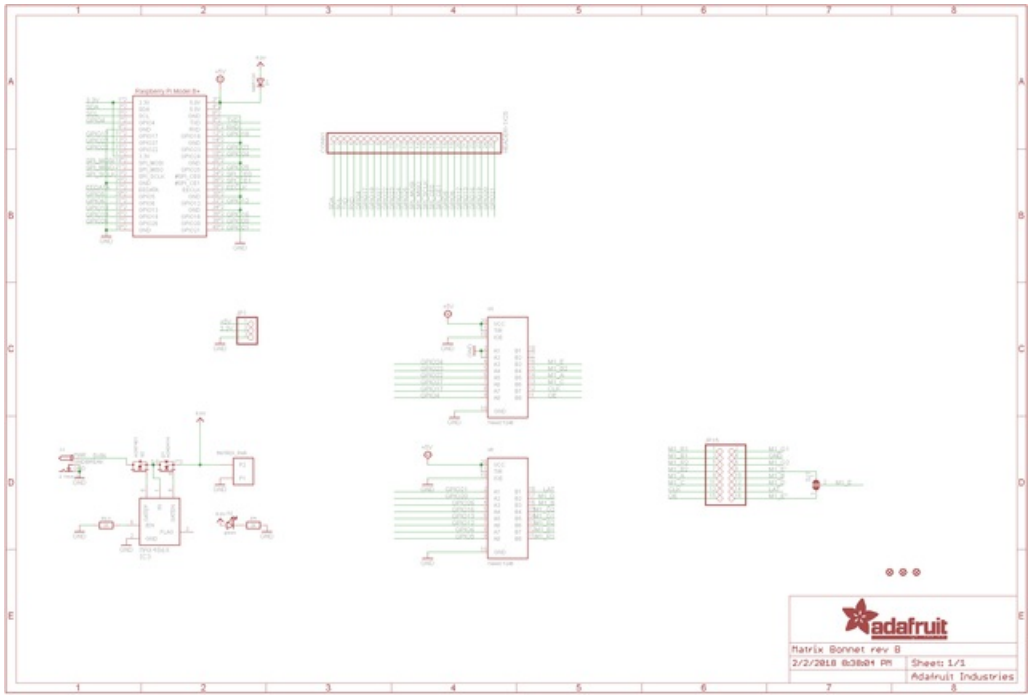
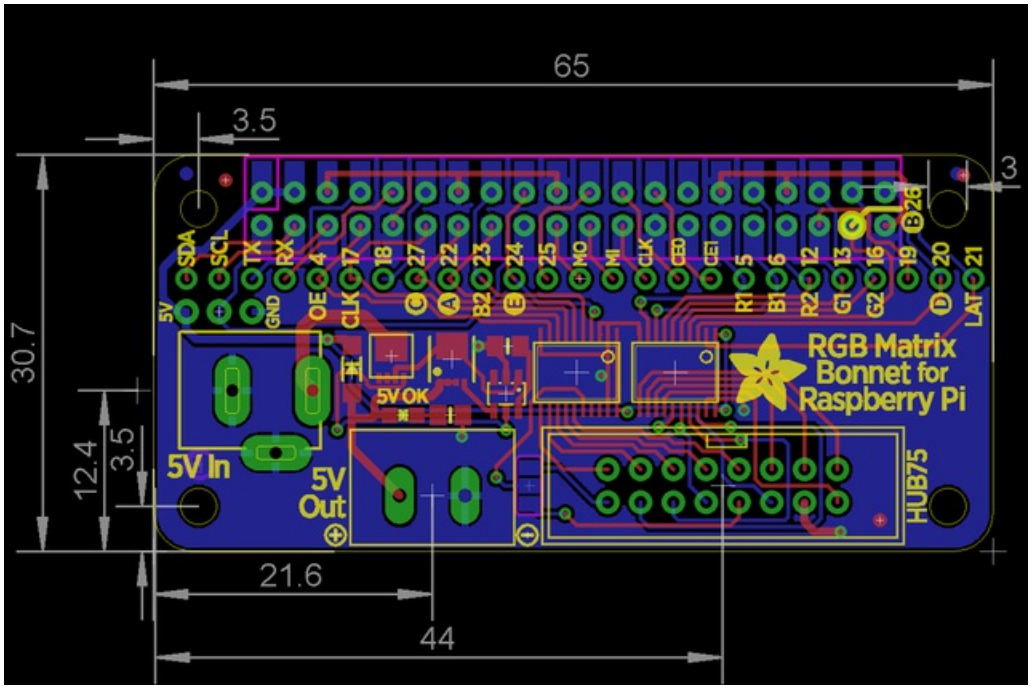
Run **sudo raspi-config** and in the “Overclock” options set the core frequency to 350 MHz or less. Reboot and see if the image is stable. There seems to be an issue when toggling GPIO too quickly.

Also see the “Driving Matrices” page for notes about dialing back the GPIO speed.

Downloads

Datasheets

- MAX4866 5V protection chip (https://adafru.it/em6)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/aP3)
- EagleCAD PCB files on GitHub (https://adafru.it/BhZ)

Schematic & Fabrication Print

Click to embiggen. Dims in mm