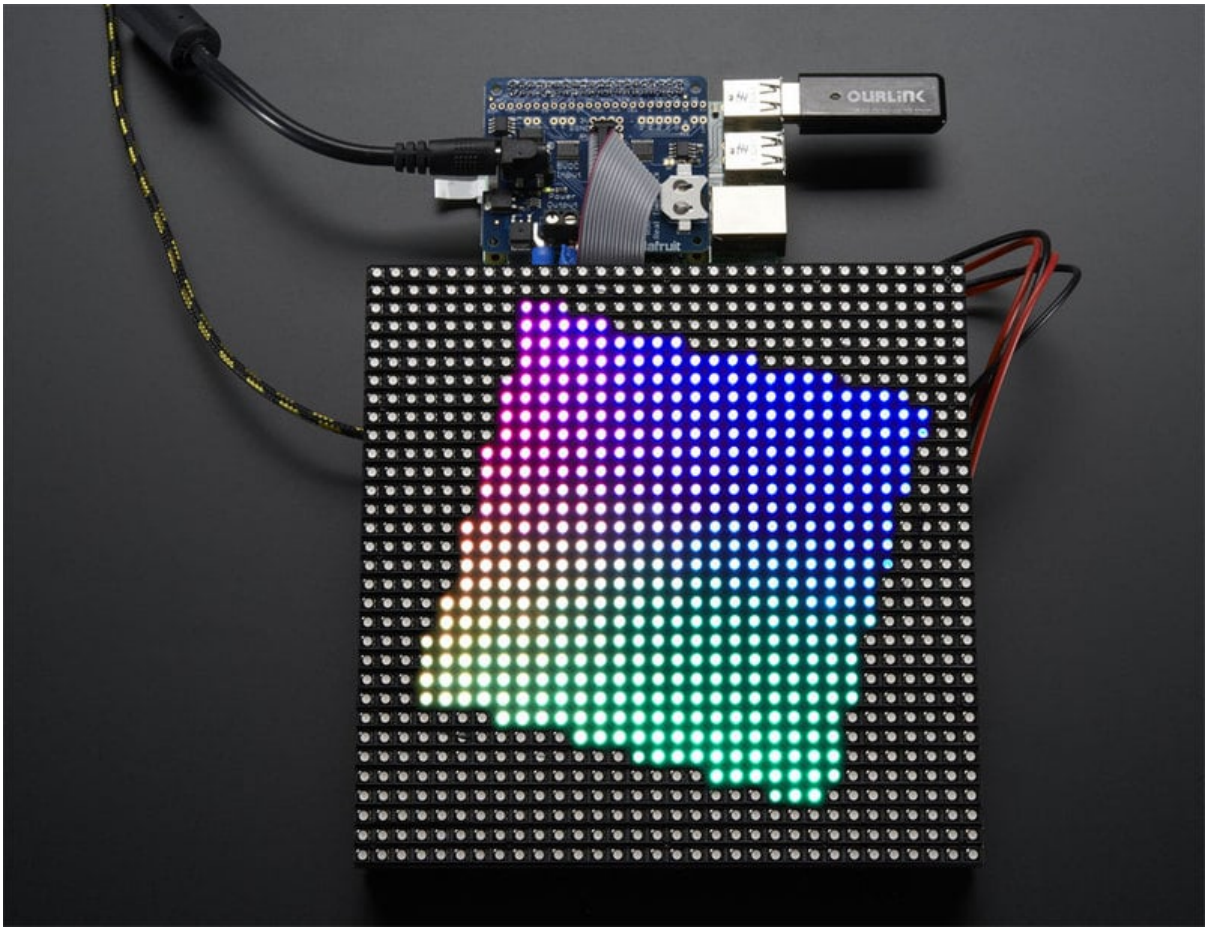
# Adafruit RGB Matrix + Real Time Clock HAT for Raspberry Pi

Created by lady ada

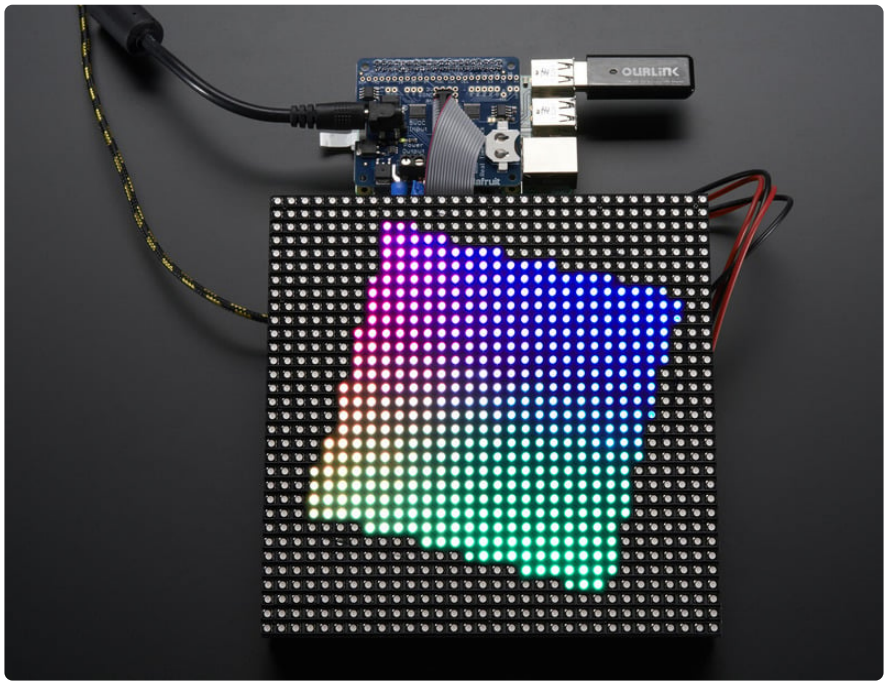


https://learn.adafruit.com/adafruit-rgb-matrix-plus-real-time-clock-hat-for-raspberry-pi

Last updated on 2025-07-16 03:38:24 PM EDT

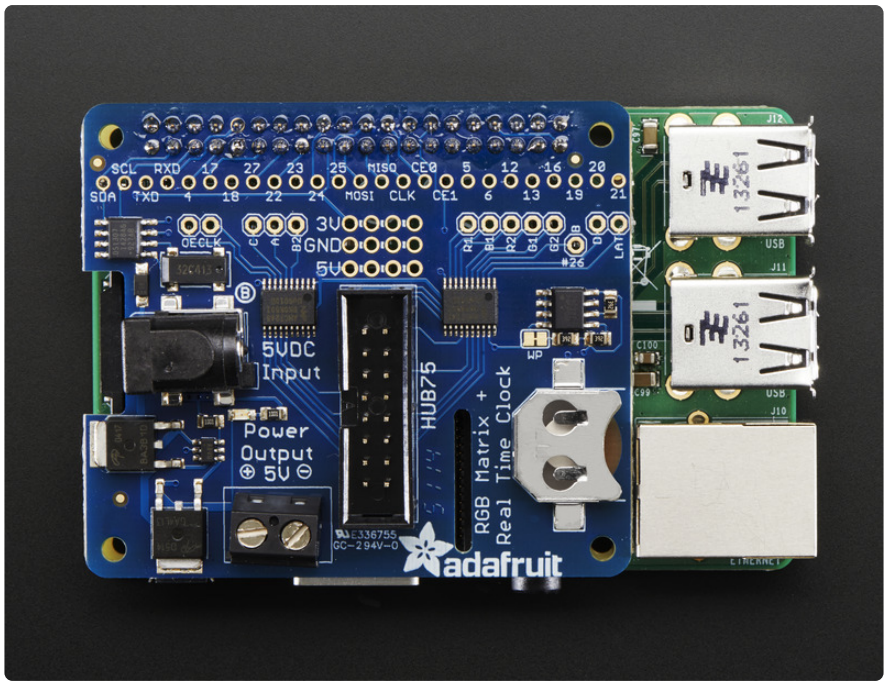
# Table of Contents

# Overview



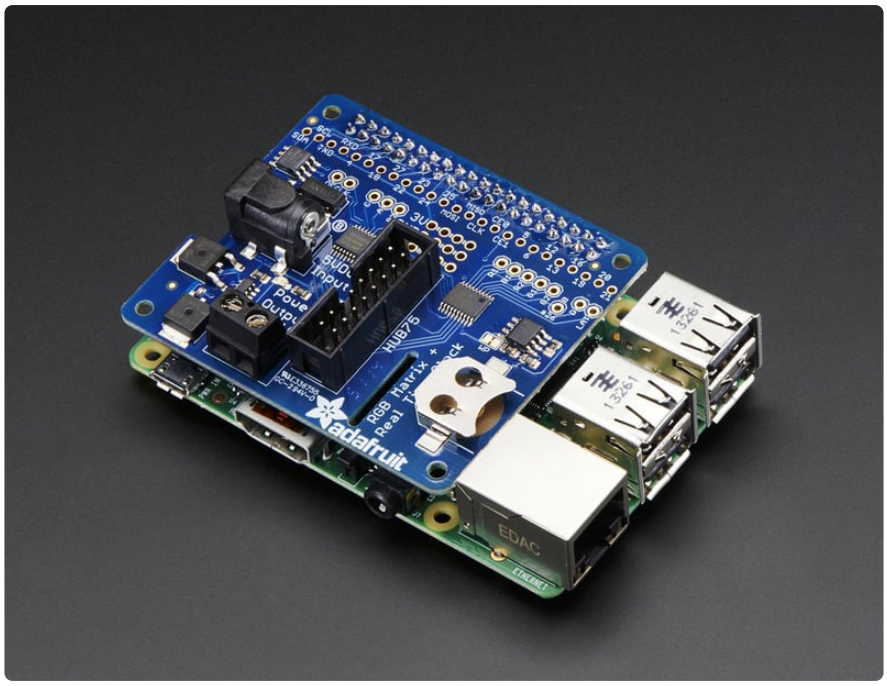
You can now create a dazzling display with your Raspberry Pi with the Adafruit RGB Matrix HAT or Bonnet. These boards plug into your Pi and makes it super easy to control RGB matrices such as those we stock in the shop and create a colorful scrolling display or mini LED wall with ease.

**The RGB Matrix HAT works on any Raspberry Pi with a 40-pin GPIO header — Zero, Zero W/WH, Zero 2 W, Model A+, B+, Pi 2, 3, and 4.** It does not work with older 26-pin boards like the original Model A or B, nor does it work with the Pi 400. Note with the Pi Zero you may need to solder a header on the Pi board; it's normally unpopulated on that model (except the "Zero WH").



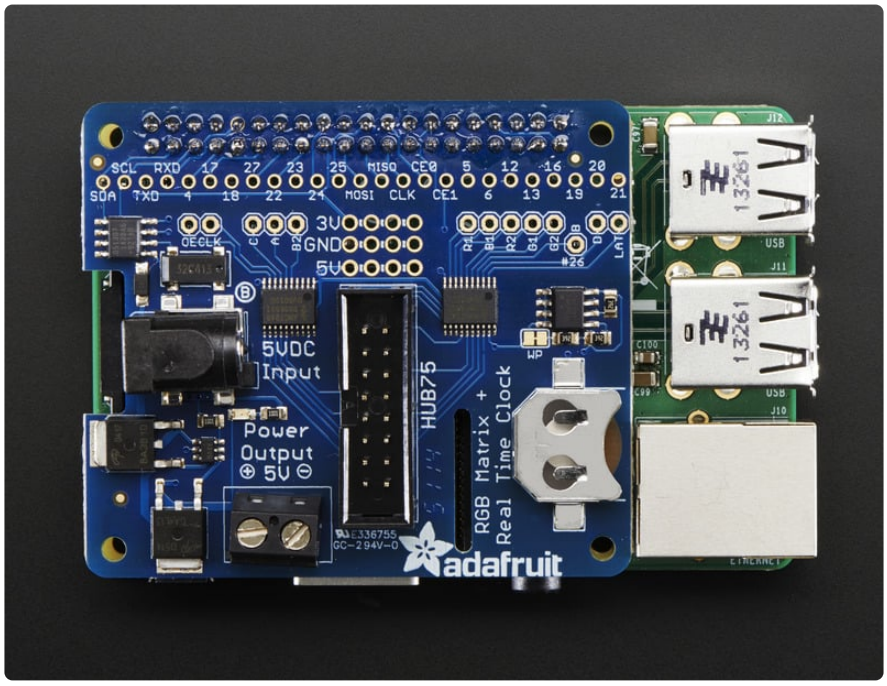

This HAT is our finest to date, full of some really great circuitry. Let me break it down for you:

- **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
- **DS1307 Real Time Clock** can keep track of time for the Pi even when it is rebooted or powered down, to make for really nice time displays




Works with any of our [16x32, 32x32 or 32x64 RGB LED Matrices with HUB75 connections (https://adafru.it/emd)](). The latest “Rev C” HAT also supports 64x64 matrices by soldering a small jumper on the PCB. You can even 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 HAT is only for use with **HUB75** type RGB Matrices. **Not for use with NeoPixel, DotStar, or other 'addressable' LEDs.**



Each order comes with a HAT PCB with all surface mount parts assembled, a 2x20 female socket connector, a 2 pin terminal block, and a 2x8 IDC socket connector. A CR1220 coin cell is not included to make air shipping easier, please order one seperately (http://adafru.it/380) if you do not have one and would like to use the real time clock.

**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 (http://adafru.it/1466).

**Raspberry Pi not included** (but we have 'em in the shop so pick one up) (https://adafru.it/eme)

Some light soldering is required to attach the headers to your Pi. A soldering iron and solder are required, but it's a simple soldering job and most beginners can do it in about 15 minutes.

---

# Pinouts

This HAT 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:** RX, TX, 25, MOSI, MISO, SCLK, CE0, CE1, 19.

Pin 24 is free if you are not using a 1/32 scan (i.e. 64x64) matrix.

Pin 18 is free if using the "convenience" (vs "quality") setting during installation.

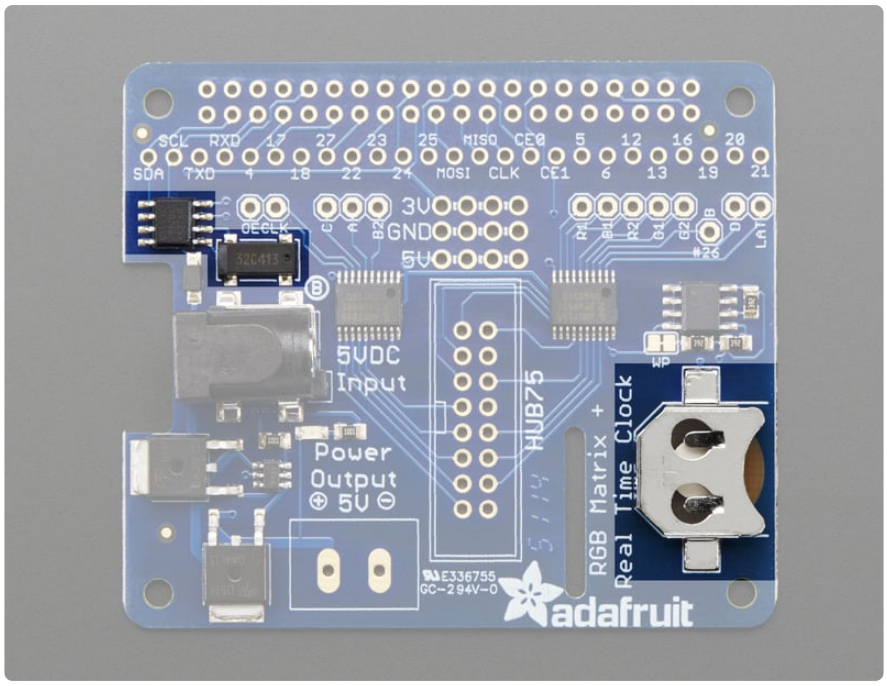
The **1-Wire** interface as enabled by raspi-config will interfere with the operation of the matrix! By default it uses pin 4. If you are connecting any 1-Wire devices, specify a **different pin** (any of the above) in /boot/config.txt, for example:

```
dtoverlay=w1-gpio gpiopin=19
```

# I2C / RTC pins

The DS1307 Real Time Clock soldered onboard is connected to the I2C pins **SDA** and **SCL** - these can still be used for other I2C sensors and devices as long as they are not on address 0x68

To use the Real Time Clock, a CR1220 3V lithium battery is required.

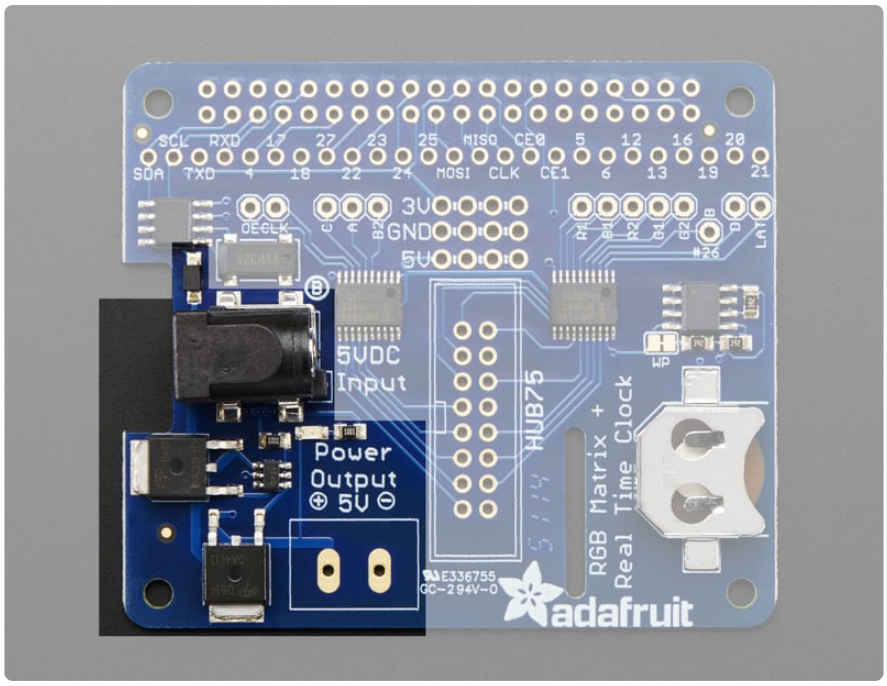



# 5V protection circuitry and backpower diode

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!

Because the lines are addressed, 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 HAT 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 HAT, 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 HAT and it will automagically power up the Pi too!

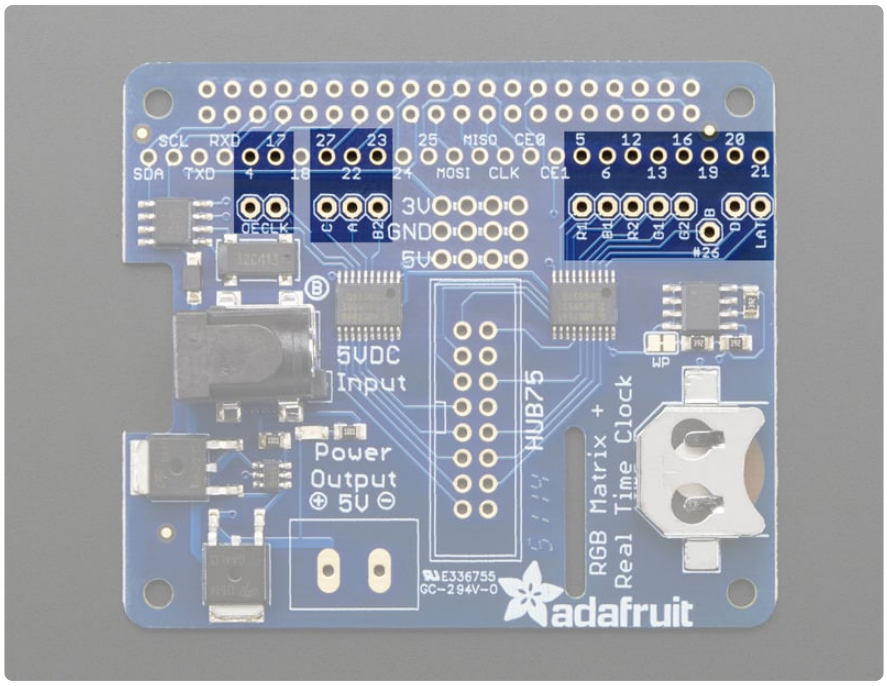
The green LED next to the DC jack will indicate that the 5V power is good, make sure it is lit when trying to use the HAT!

# 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 74AHCT145 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

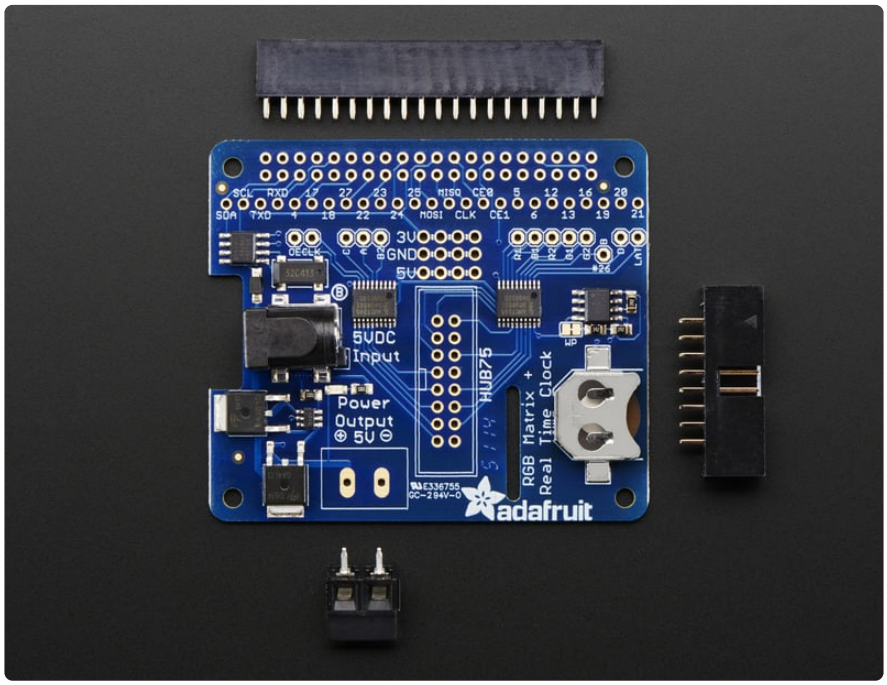
## RGB Matrix Address pins

- Pi GPIO **#22 - Matrix A** (address A) pin
  This pin is part of the 1->16 or 1->8 multiplexing circuitry.

- Pi GPIO **#26 - Matrix B** (address B) pin
  This pin is part of the 1->16 or 1->8 multiplexing circuitry.
- Pi GPIO **#27 - Matrix C** (address C) pin
  This pin is part of the 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. **Present on newer "Rev C" HATs only. Requires minor soldering, explained on next page.**

---

# Assembly

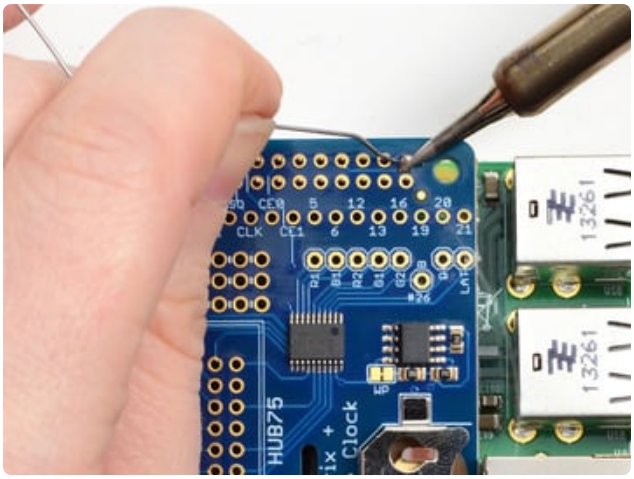


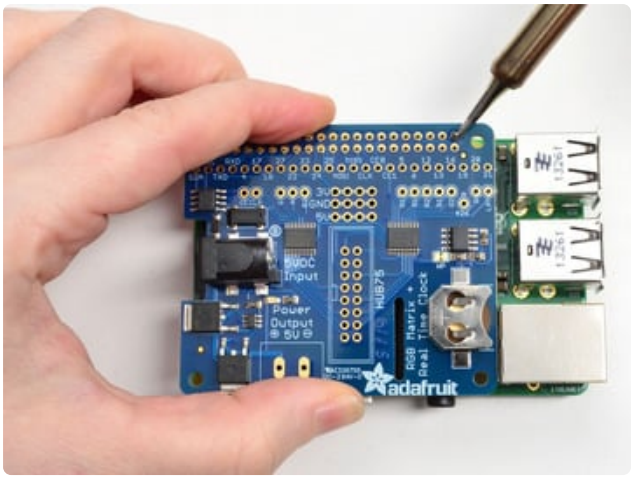

## Solder on Headers and Terminal Block

Before we can a-blinkin' there's a little soldering to be done. This step will attach the 2x20 socket header so that we can plug this HAT into a Raspberry Pi, the 2x8 header so we can plug the RGB matrix into the HAT, and a terminal block so you can power the matrix through the HAT.

Start by plugging the 2x20 header into a Raspberry Pi, this will keep the header stable while you solder. Make sure the Pi is powered off!


Place the HAT on top so that the short pins of the 2x20 header line up with the pads on the HAT



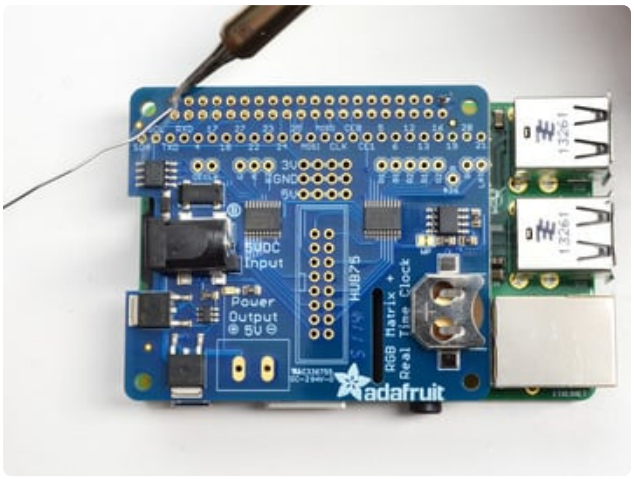


## And Solder!

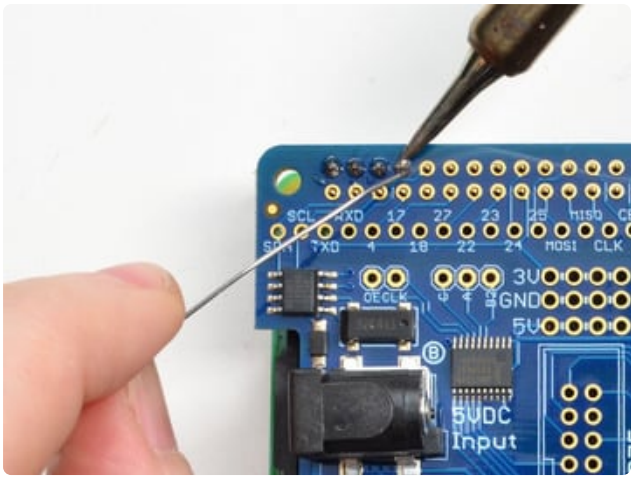
Heat up your iron and solder in one header connection on the right.

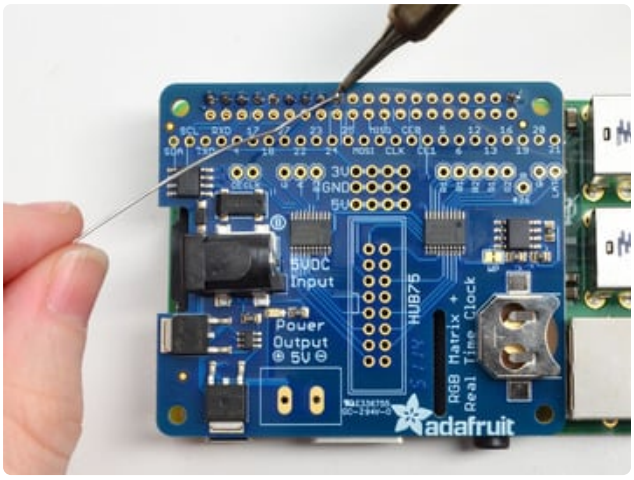
Once it is soldered, put down the solder and reheat the solder point with your iron while straightening the HAT so it isn't leaning down

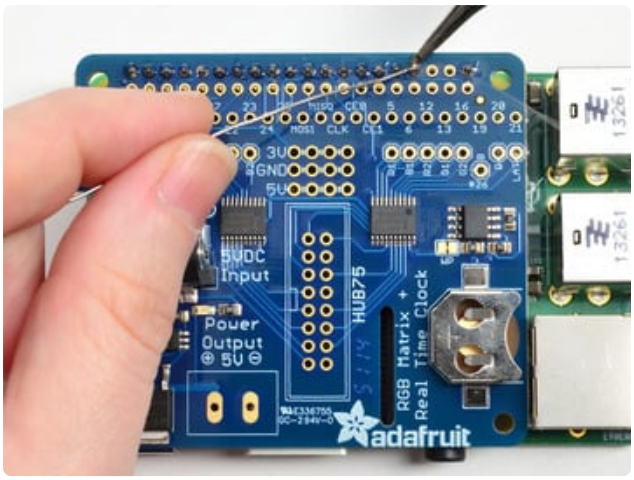
(For tips on soldering, be sure to check out our Guide to Excellent Soldering (https://adafru.it/aTk)).



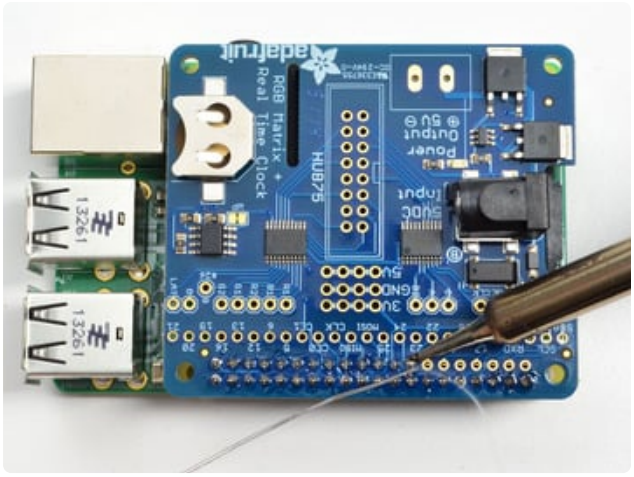
Solder one point on the opposite side of the connector

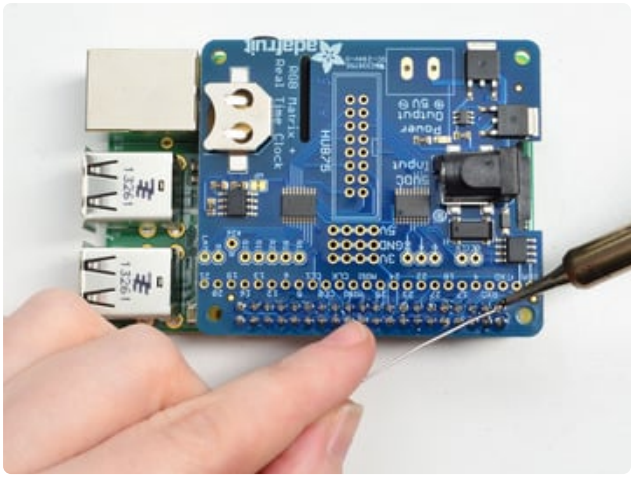


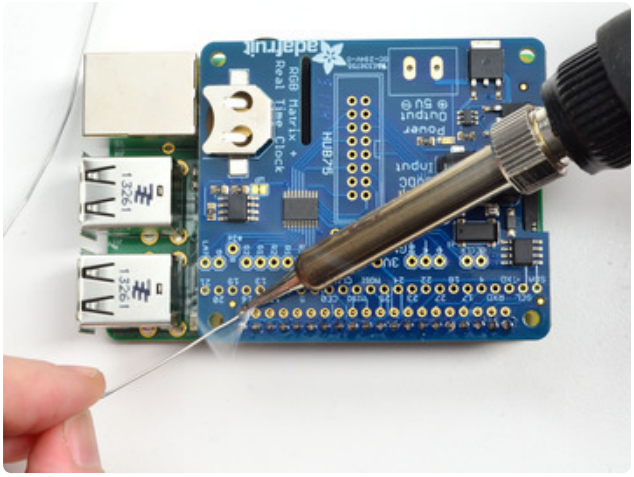



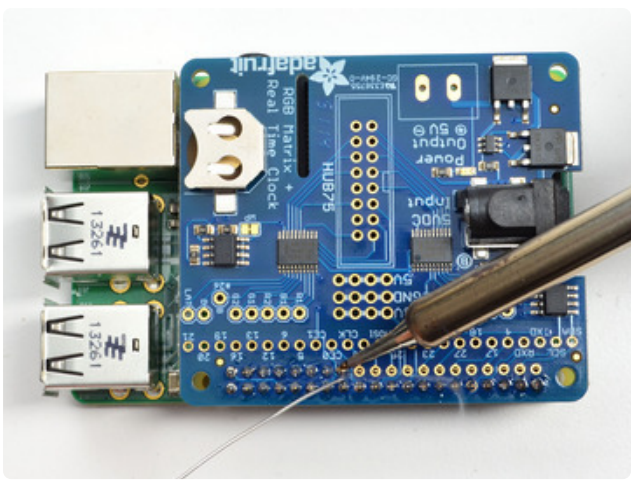

Solder each of the connections for the top row

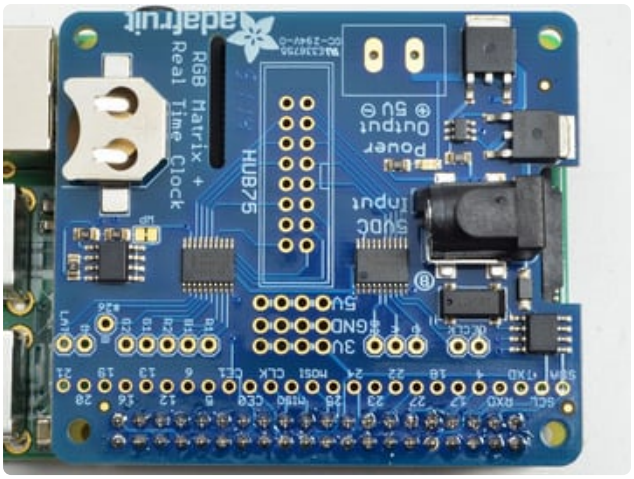




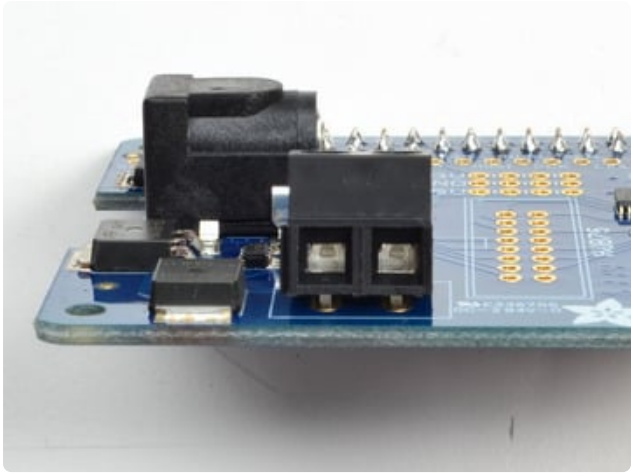
Flip the board around and solder all the connections for the other half of the 2x20 header



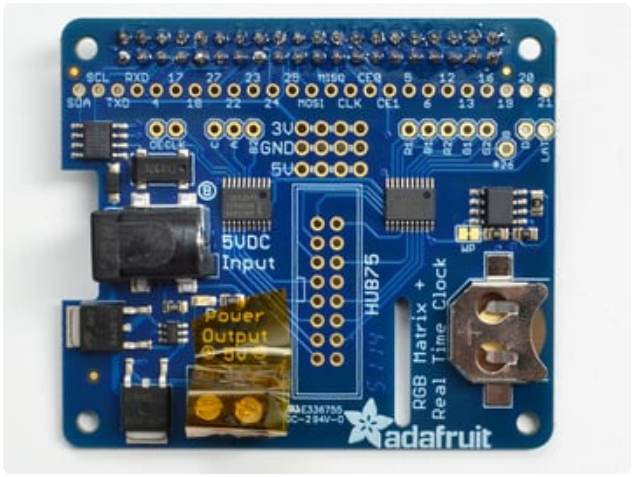




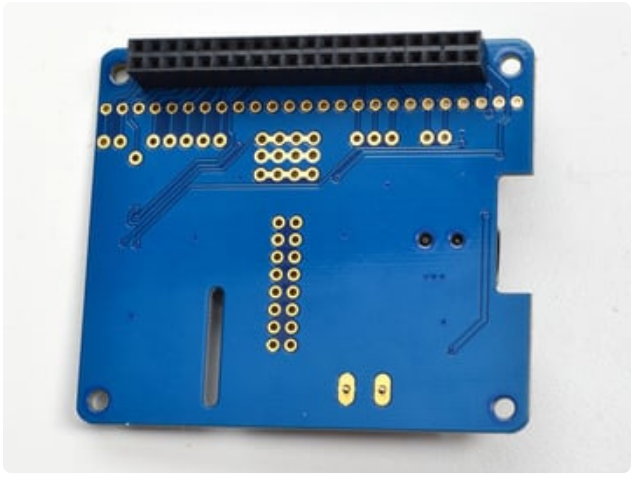

Check over your work so far, make sure each solder point is shiny, and isn't bridged or dull or cracked



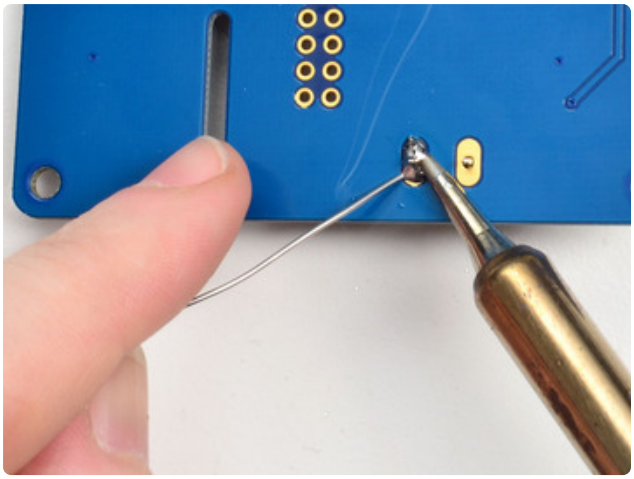
Place the 2 pin terminal block first, make sure the two 'mouths' are facing outwards

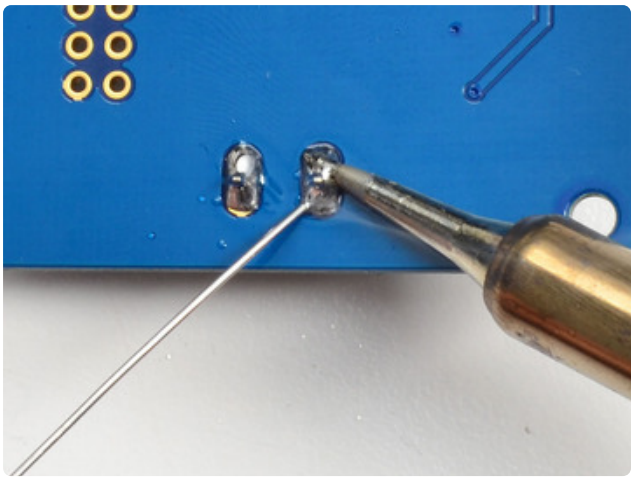



Use some tape to stick the terminal down in place

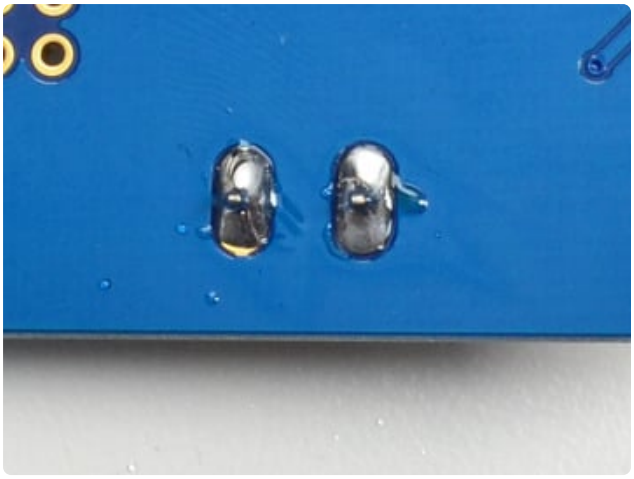


Flip the board over, the tape should keep the terminal block in place



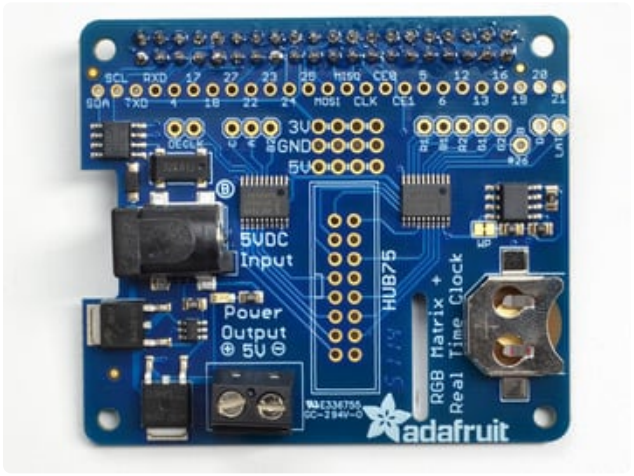


Solder the two big connections, use plenty of solder!



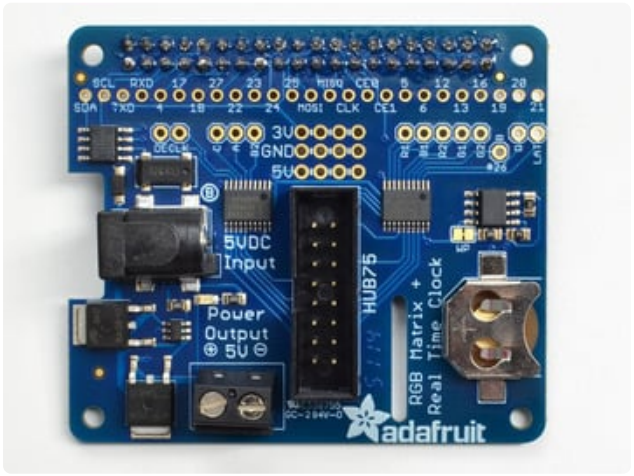
Check your work, the connections should be solid and shiny

Next up we will attach the 2x8 IDC header. Unlike the 2x20 header, **this connector has a direction!**

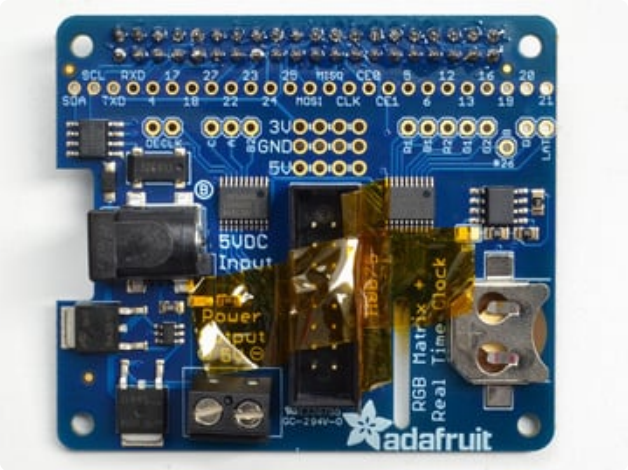



Notice in the middle there's an outline for the connector in the middle. On the right it says **HUB75** and on the left of the connector there is a little 'cutout' shape. This cutout shape must match up with the cut out on the connector.

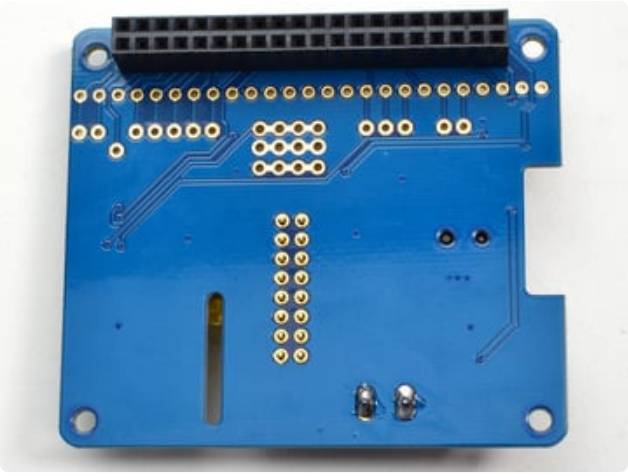
If you solder it in backwards, its not a huge deal, you can use diagonal cutters to cut out a notch on the opposite side, but if you get it right then you will never have to worry about plugging in your matrix data cable the wrong way



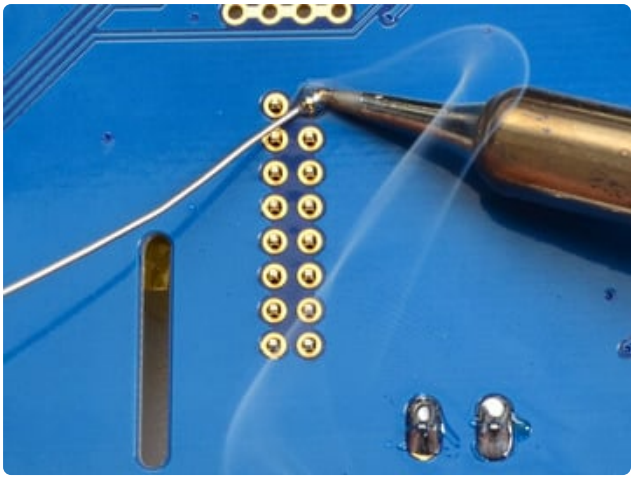

Place the connector in the slot so that the notched side is on the **left**

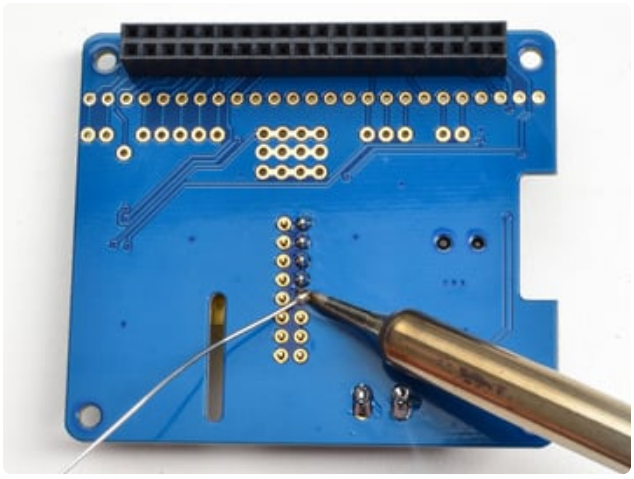



Use some tape to hold the IDC connector in place

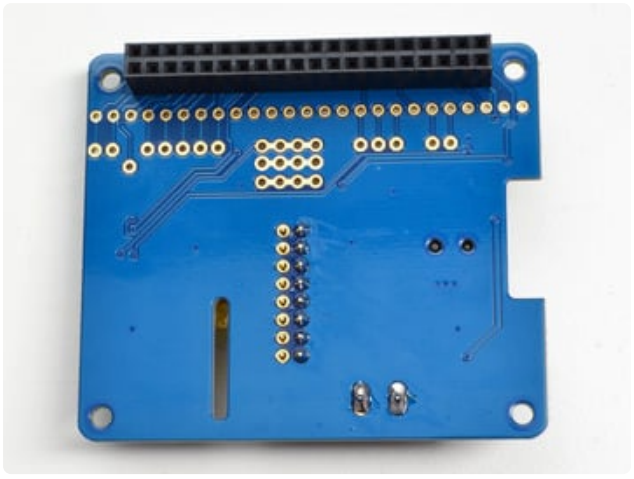


Flip the board over, the tape should keep the connector from falling out

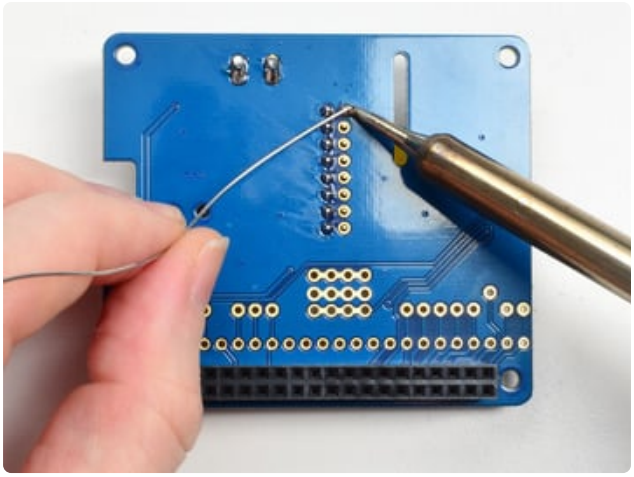




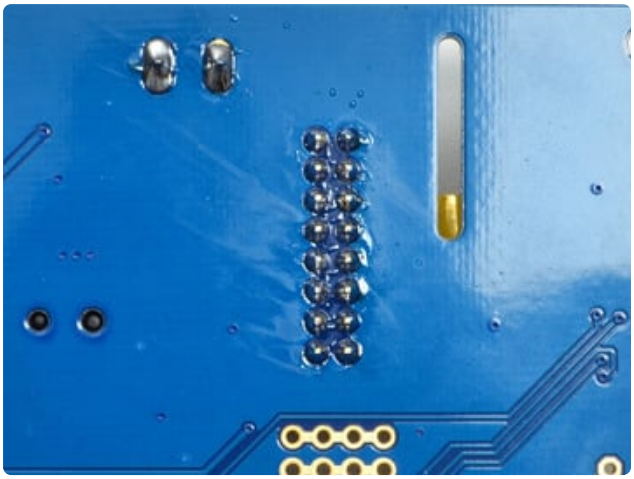
Solder in all the pins like you did with the 2x20 connector



Check your work! Make sure all the solder points are clean and not shorted or cracked or dull



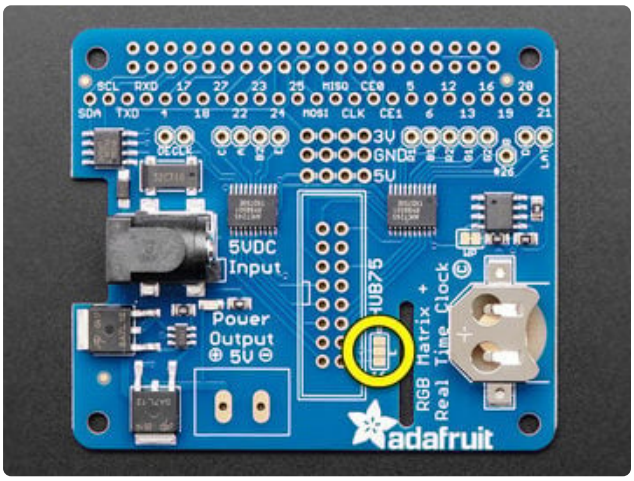
Flip the board around & solder up the other half!



Check your work one last time...now continue to testing!

# 64x64 Matrices: Solder “E” Jumper

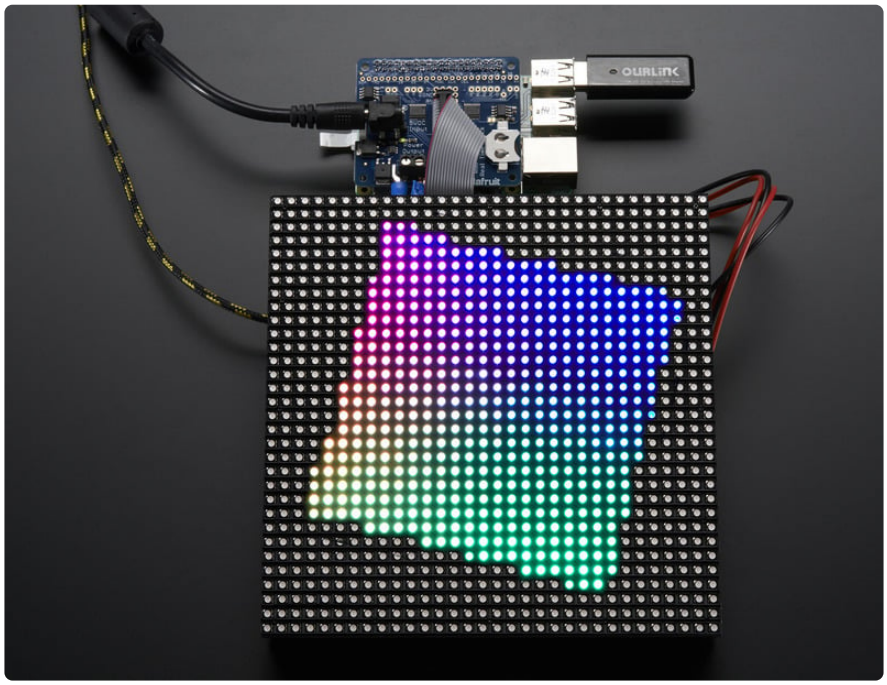
**64x64 matrices are supported on the latest “Rev C” HATs only.**




Look for the Address E pads located between the HUB75 connector and Pi camera cutout.

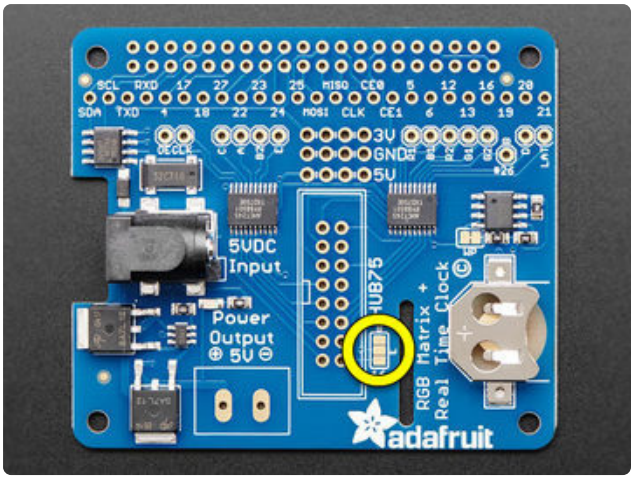
Melt a blob of solder between the center “E” pad and the “8” pad just above it (for 64x64 matrices in the Adafruit shop)...or the “16” pad below (rare, for some third-party 64x64 matrices...check datasheet).

# Driving Matrices



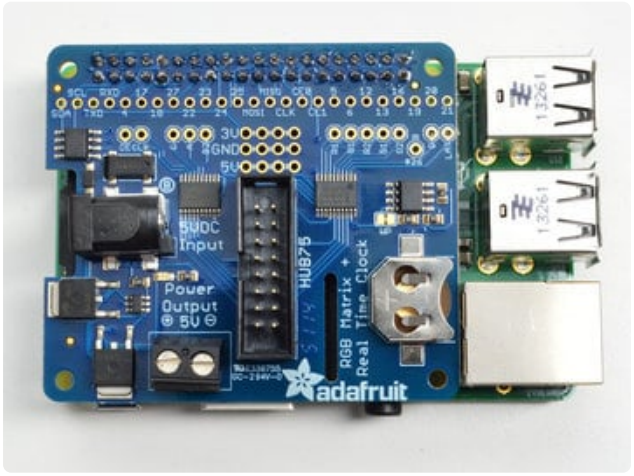
OK we're onto the fun part now! Be sure you have completed the Assembly step before continuing, the soldering is **not optional.**

Before connecting any cables, however, it makes sense to get the soldering out of the way if needed.

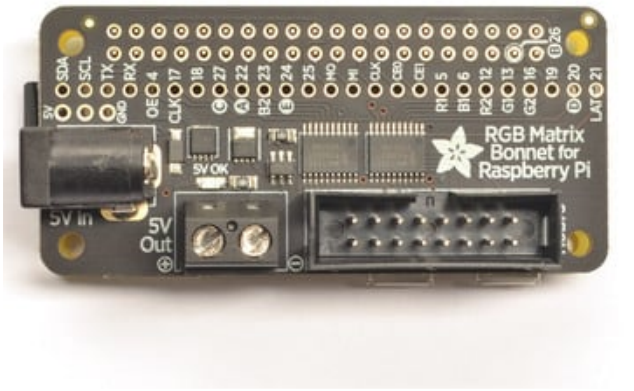




If you're using a **64x64** RGB matrix and either a Bonnet or a Rev C HAT, use your soldering iron to melt a blob of solder on the bottom solder jumper so the middle pad is 'shorted' to **8**. (This is compatible with 64x64 matrices in the Adafruit store. For 64x64 matrices from other sources, you might need to use 16 instead, check the datasheet.)

# Step 1. Plug HAT/Bonnet into Raspberry Pi

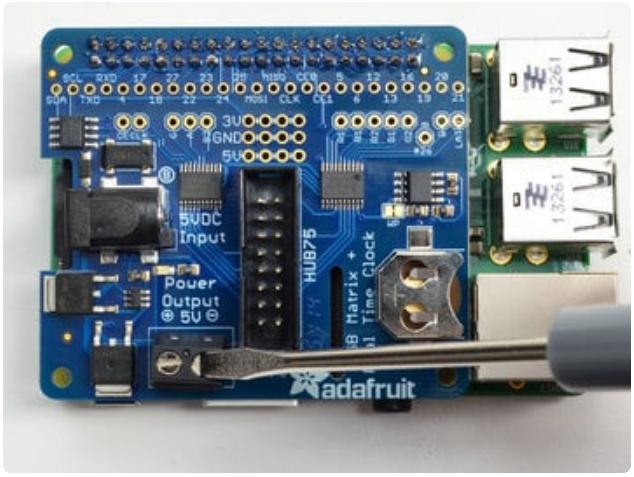



Shut down your Pi and remove power. Plug the HAT or Bonnet on so all the 2x20 pins go into the GPIO header.

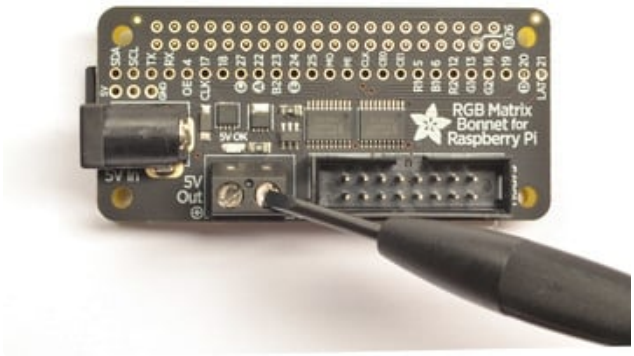


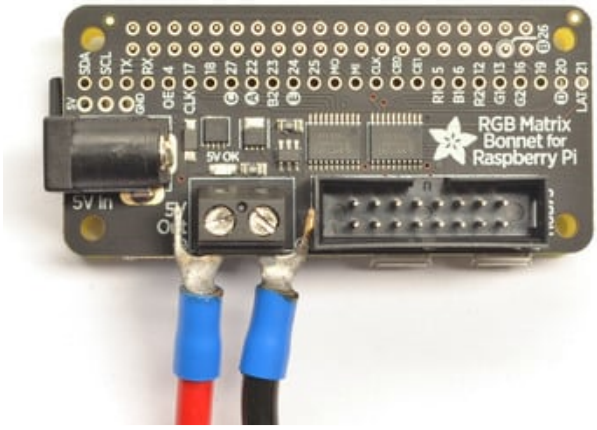

# Step 2. Connect Matrix Power cable to terminal block

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



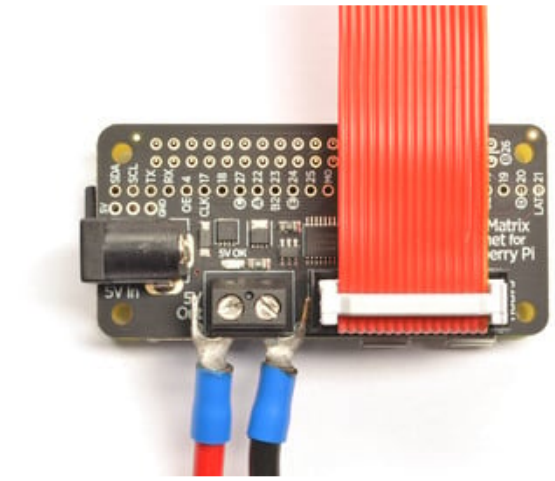

Either way, unscrew the terminal blocks to loosen them




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.

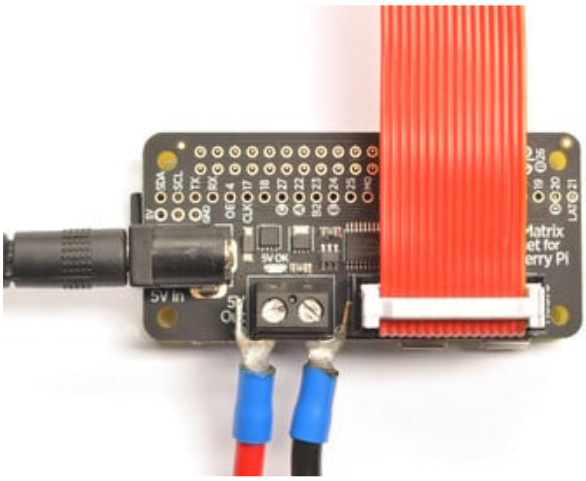
It wont damage the matrix if you accidentally get the cable connected to the output end of the matrix but it wont work so you might as well get it right first time!

# 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.

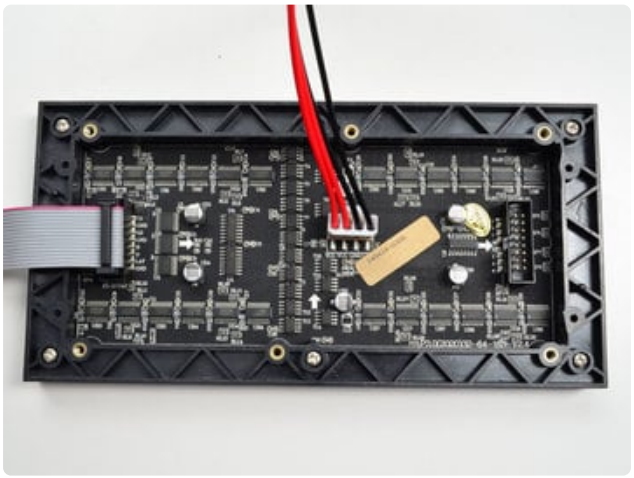
You **can** power the Pi via the 5V wall plug that is also used for the Matrix but its best to have it powered seperately

# 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 "Buster" (or earlier "Stretch") operating system** (either the Lite or Desktop version):

At this time, the LED Matrix library does not work on the Pi 5, nor the Pi 400.

```
curl https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-Scripts/main/
rgb-matrix.sh &gt;rgb-matrix.sh
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.

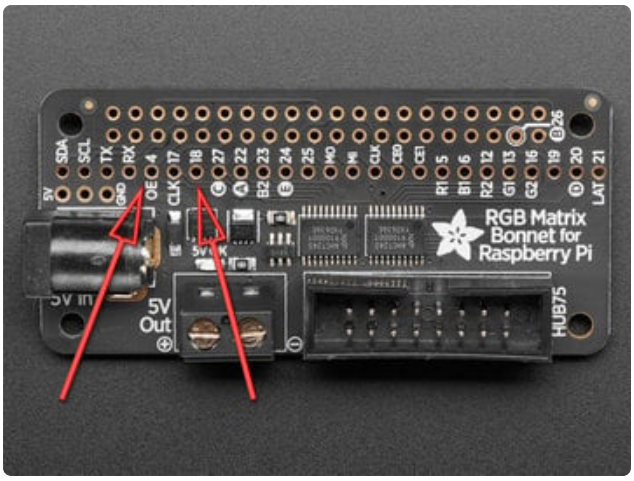
Of particular note here: any existing installation will be replaced. If there is a directory called "rpi-rgb-led-matrix" in the current working directory, its contents will be overwritten. Additionally, a Python module is installed and will replace anything currently there. If this is a problem, cancel 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:
```

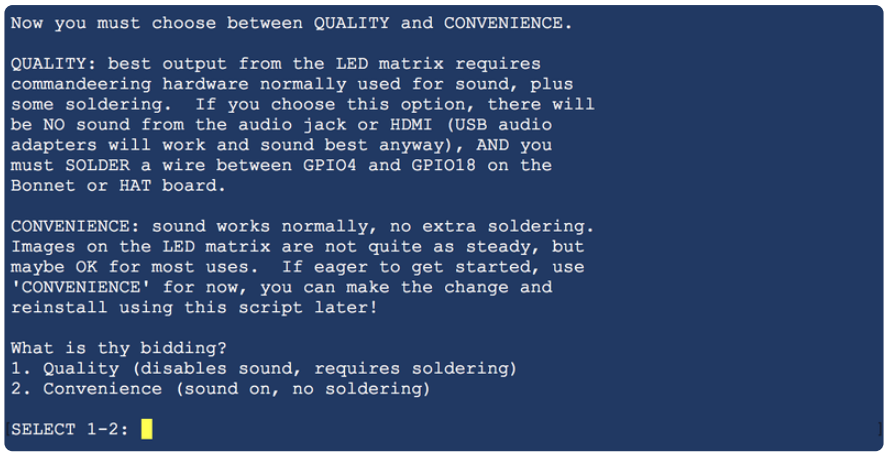
Then you're asked whether you need the absolute best image possible from the LED matrix, or can accept slightly reduced quality for the sake of simplicity.




**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 an occasional 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=16 --led-cols=32
```

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...n) Sometimes needed to throttle back the speed when using a fast Pi. Default is **1**.

**For Raspberry Pi 3 use a slowdown of 1 to start (use higher values if image still flickers). For Raspberry Pi 4, use a slowdown of 4. Older Pi models might work with 0, try it.**

`--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 or Pi 4, you may need to dial back the matrix control speed slightly. This can be done with the `--led-slowdown-gpio=2` setting. Pi 4 may require larger values, depending on the matrix...**experiment!** 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 Python 3 and for Python 2.7 if present (the latest Raspberry Pi OS versions no longer include Python 2.7, but it can be optionally installed it needed.

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.

To run the python examples, you can use the same parameters above to the command, though there is an additional one that you will need to provide:

`--led-gpio-mapping=` (hardware) For either theRGB Matrix HAT or bonnet, you will want to choose **adafruit-hat** if you installed with the convenience option or **adafruit-hat-pwm** if you installed with the quality option. For example, to run the text scrolling sample on a single 64x32 RGB panel connected to a Pi 4 and installed with the convenience option, you would run:

```
sudo python runtext.py -m=adafruit-hat --led-rows=32 --led-cols=64 --led-slowdown-gpio=4
```

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.

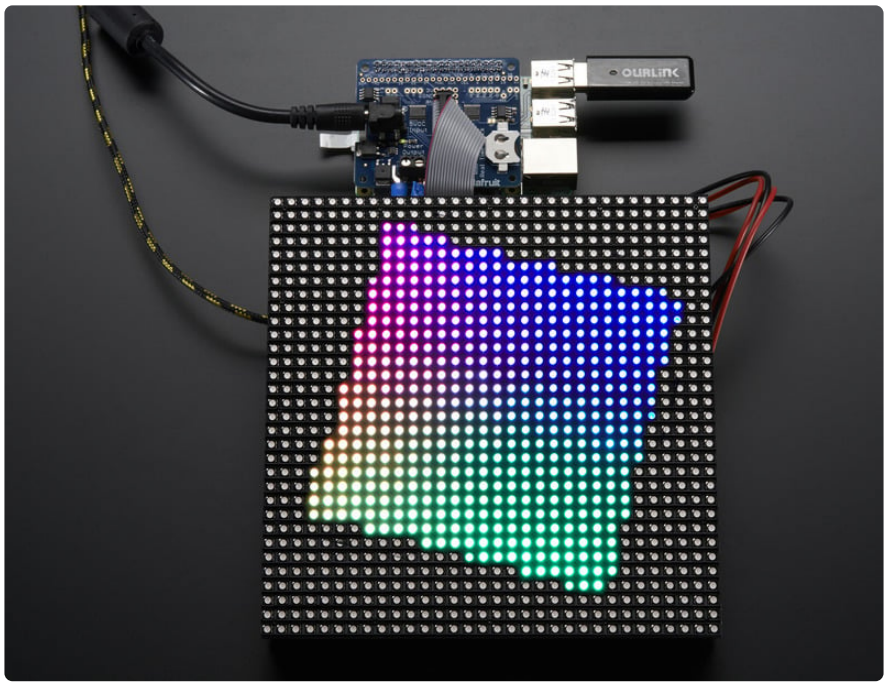
**Core PIL image functions are explained here: The Image Module (https://adafru.it/dvE)**

**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!**

---

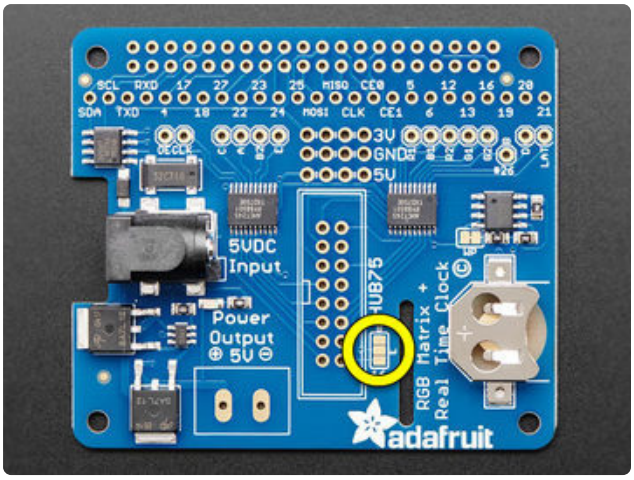
# Matrix Setup



## Configure for 64x64 Matrix

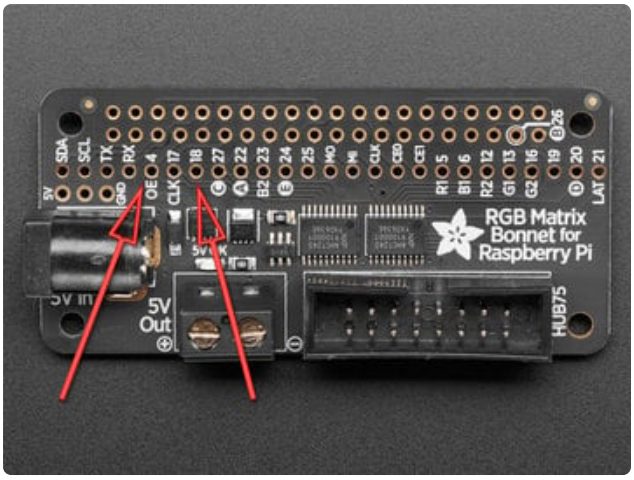
Additional soldering is needed if using a **64x64** RGB Matrix.


If you're using a **64x64** RGB matrix and either a Bonnet or a Rev C HAT, use your soldering iron to melt a blob of solder on the bottom solder jumper so the middle pad is 'shorted' to **8**. (This is compatible with 64x64 matrices in the Adafruit store. For 64x64 matrices from other sources, you might need to use 16 instead, check the datasheet.)




## Configure for Quality/Convenience

This an option picked when running the script to install the software. No additional work is needed if the "convenience" option is chosen. For the "quality" option, a connection is needed between **GPIO4** and **GPIO18**.

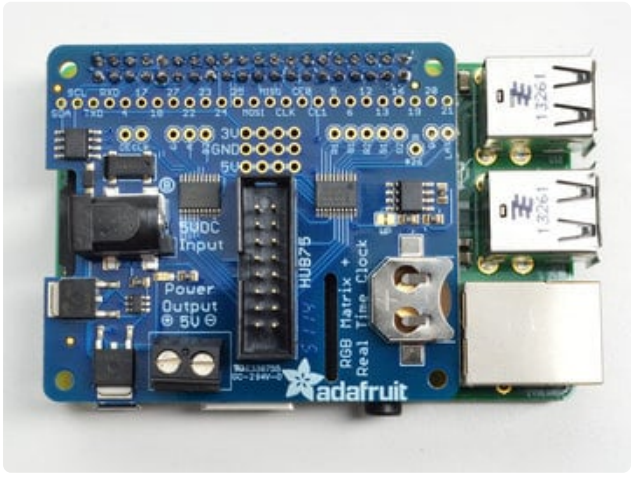



**The “quality” option comes at a cost.** First, you need to **solder a jumper wire** between **GPIO4** and **GPIO18** on the Bonnet or HAT board.

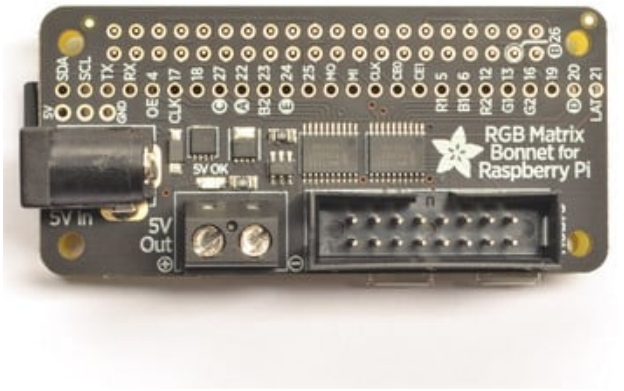
Additionally, normal audio output needs to be disabled if "quality" is chosen. Specifically, it is the `snd_bcm2835` kernel module that needs to be disabled. The installer script will take care of that if the "quality" option is chosen. USB audio

adapters should still work. But **audio over HDMI or from the 1/8" audio jack will not be present.**

# Step 1. Plug HAT/Bonnet into Raspberry Pi

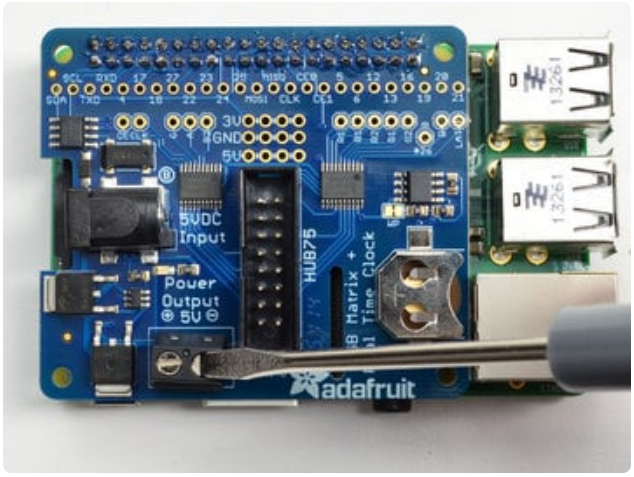



Shut down your Pi and remove power. Plug the HAT or Bonnet on so all the 2x20 pins go into the GPIO header.

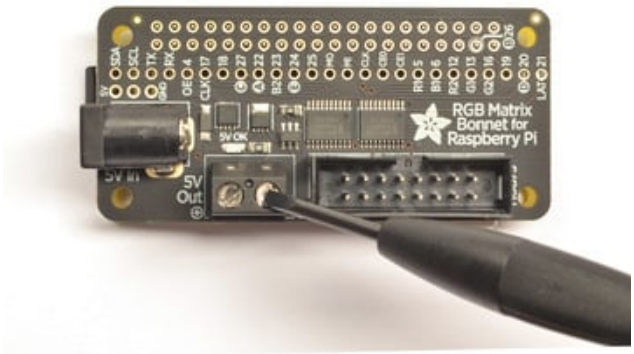



# Step 2. Connect Matrix Power cable to terminal block

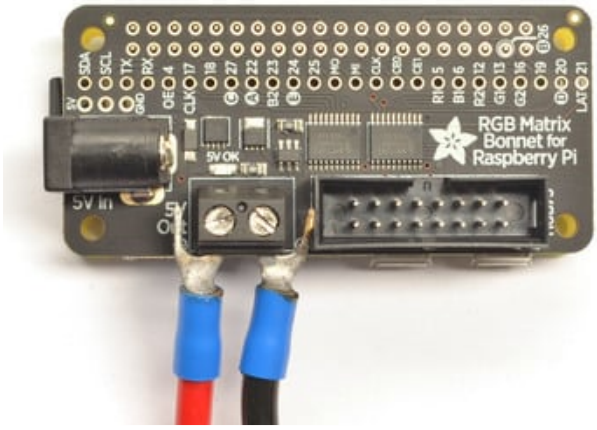
Your RGB matrix came with a red & black power cable. One end has a 4-pin MOLEX connector that goes into the matrix. The other end probably has a spade connector. If you didn't get a spade connector, you may have to cut off the connector and tin the wires to plug them into the terminal block



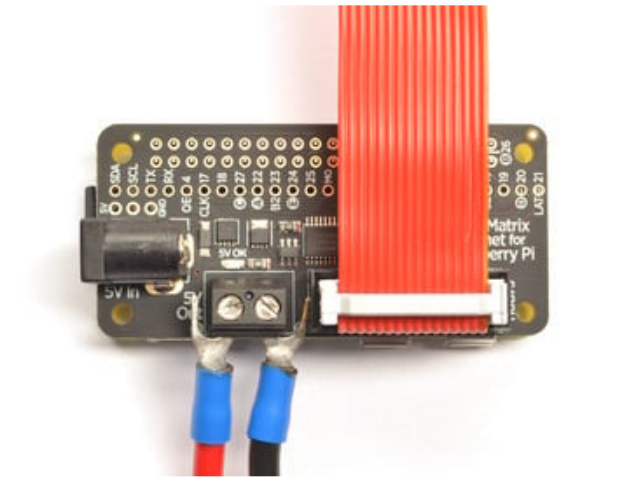

Either way, unscrew the terminal blocks to loosen them




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.

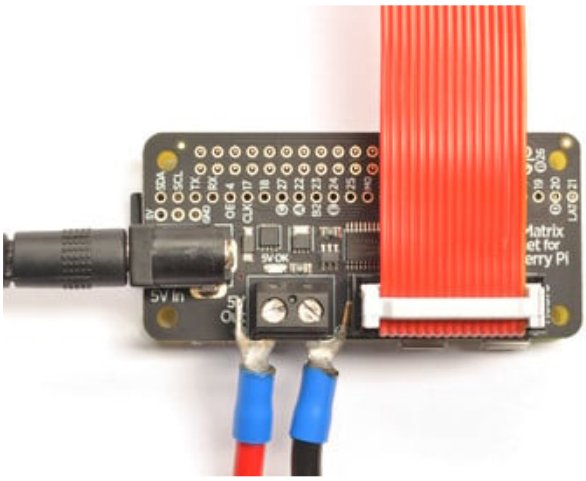
It wont damage the matrix if you accidentally get the cable connected to the output end of the matrix but it wont work so you might as well get it right first time!

# 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.

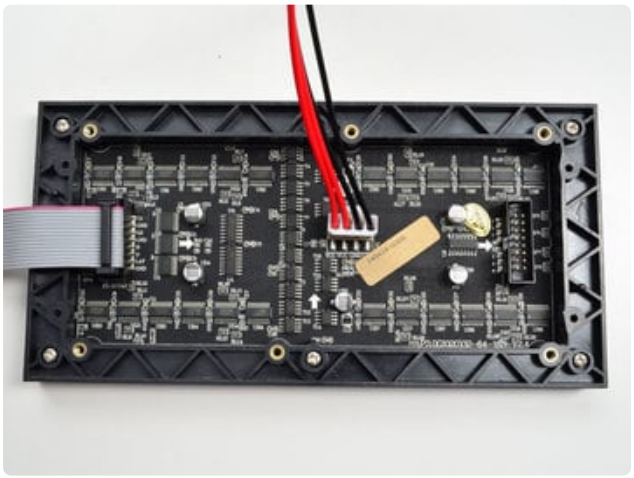
You **can** power the Pi via the 5V wall plug that is also used for the Matrix but its best to have it powered seperately

# 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 install the RGB matrix driver 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.

At this time, the LED Matrix library does not work on the Pi 5, nor the Pi 400.

**For Raspberry Pi 5s see the following guide for software setup:**

**RGB Matrix Panels With Raspberry Pi 5**

https://adafru.it/1ag4

**For Raspberry Pi 4s and older models, continue with this guide to install software.** There is an automated script that can be run. For people more comfortable with cloning repos and running make files, a manual install process is also shown. Pick whichever one seems best suited for your setup.

---

# Install Using Script

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.

From the command line, use the following commands to run the installer script:

```
curl https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-Scripts/main/
rgb-matrix.sh &gt;rgb-matrix.sh
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. 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. If you are using the Adafruit Triple LED Matrix Bonnet then select **Adafruit RGB Matrix Bonnet**.

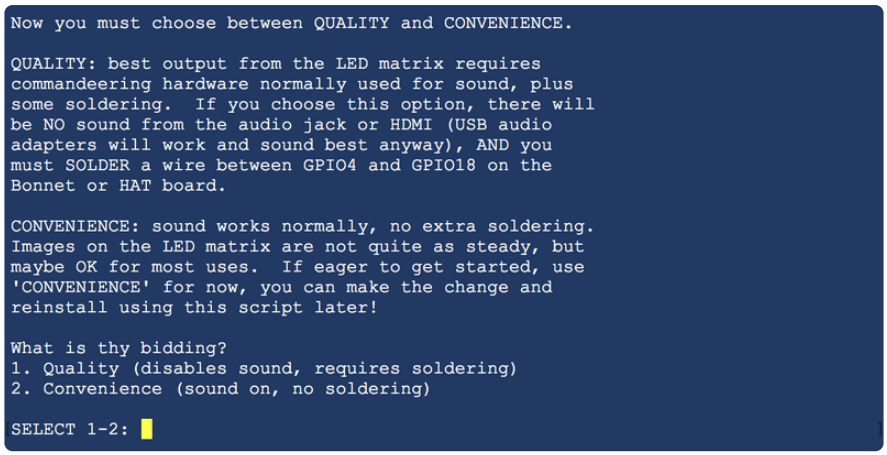
```
Select interface board type:
1. Adafruit RGB Matrix Bonnet
2. Adafruit RGB Matrix HAT + RTC

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" setting requires soldering the GPIO4 and GPIO18 pins together.** See previous section for details. If you are using the **Adafruit Triple LED Matrix Bonnet** you should select **"quality"**. The **Adafruit Triple LED Matrix Bonnet** does not require any additional soldering. The single matrix bonnet and HAT require additional soldering for the quality choice.

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

If you're using a single matrix bonnet or HAT, and 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

Once the Pi has rebooted, log back in and continue to the **Testing Install** section.

---

# Install Manually

> This information is for advanced users. It is recommended to use the installer script from the previous section.

## Install Prerequisites

Make sure these are installed:

```
sudo apt-get install -y git python3-dev python3-pillow cython3
```

## Source Code

The code used to drive the matrices is found in this repo:

**rpi-rgb-led-matrix**

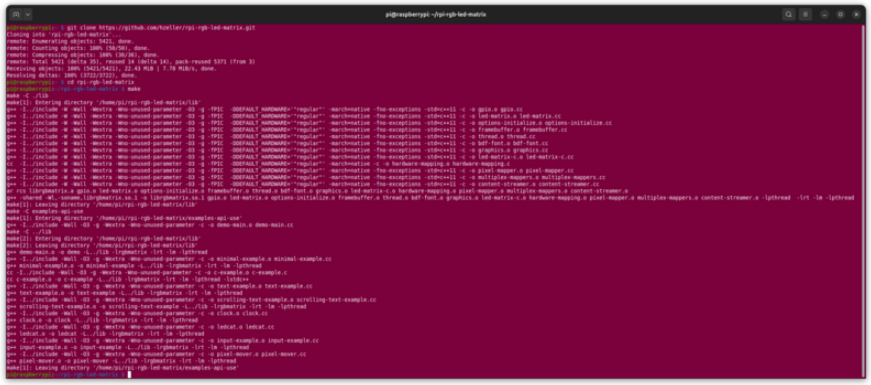
https://adafru.it/kdg

There is good information found there in various README's and that might be all you need. The information that follows serves more as a general example of how to get the code and build the main targets.

## Clone and Build

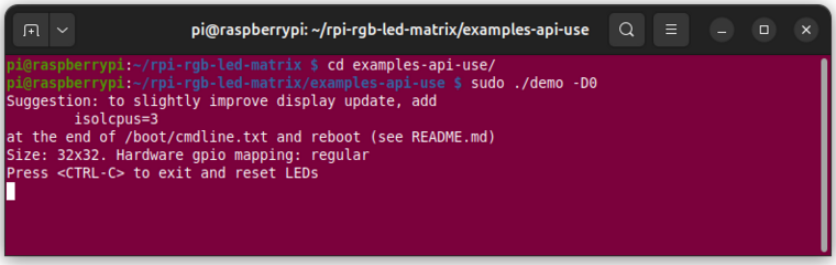
The project is makefile based, so it is fairly easy to clone the repo and build the target(s):

```
git clone https://github.com/hzeller/rpi-rgb-led-matrix.git
cd rpi-rgb-led-matrix
make
```



The default target builds the library as well as the C examples found in the `examples-api-use` subdirectory. It should be possible to run those examples after the build completes:

```
cd examples-api-use
sudo ./demo -D 0
```



## Dealing with "quality" vs. "convenience"

The default build uses a hardware subsystem on the Pi that is also used for audio on HDMI and the 1/8" jack. This is the "quality" option referred to by the installer script and it also requires connecting GPIO 4 to 18 on the HAT/Bonnet. **So there is a hardware resource conflict that must be resolved.** The demo programs will report this with the following message:

```
=== snd_bcm2835: found that the Pi sound module is loaded. ===
Don't use the built-in sound of the Pi together with this lib; it is known to be
incompatible and cause trouble and hangs (you can still use external USB sound
adapters).

See Troubleshooting section in README how to disable the sound module.
You can also run with --led-no-hardware-pulse to avoid the incompatibility,
but you will have more flicker.
Exiting; fix the above first or use --led-no-hardware-pulse
```

The other option is to disable the hardware pulsing feature of the RGB matrix driver software. This is the "convenience" option referred to by the installer script.

More info on these two options:

### Option 1: Disable Sound for Hardware Pulsing ("quality")

To disable the Pi audio, blacklist it. Create a file named **/etc/modprobe.d/blacklist-rgb-matrix.conf** with the contents:

```
blacklist snd_bcm2835
```

and then reboot the Pi.

### Option 2: Disable Hardware Pulsing ("convenience")

The command line parameter `--led-no-hardware-pulse` can be used to turn off this feature in the RGB matrix driver software at run time:

```
sudo ./demo --led-no-hardware-pulse -D 0
```

It can also be disabled at build time using the DISABLE_HARDWARE_PULSES preprocessor directive:

```
make USER_DEFINES="-DDISABLE_HARDWARE_PULSES"
```

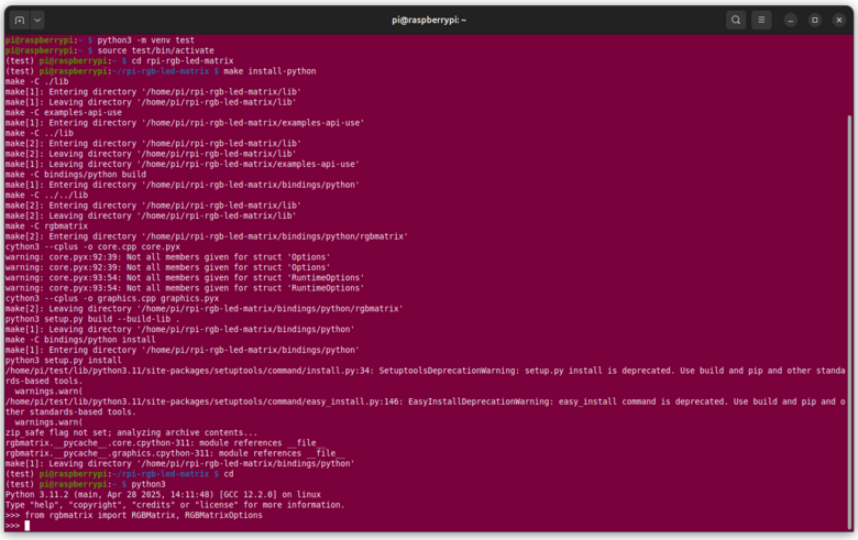
## Python Build

There are two targets for this: `build-python` and `install-python`. The install target uses an older (setup.py) approach. However, it seems to generally work OK to

create and activate a Python virtual environment and then use the `install-python` target:

> Make is run here without sudo since we are installing into a local user directory, i.e. the venv.

```
python3 -m venv test
source test/bin/activate
cd rpi-rgb-led-matrix
make install-python
```




It also works to use the `build-python` target and then manually run setup.py with a virtual environment active. The `build-python` and `install-python` targets are dependent on the main library target, so if the goal is to have Python bindings, either of those can be used as the build target, i.e. just build everything in one make execution.

---

# Testing Install

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.

These are compiled C based stand alone examples that do not rely on Python. There's no need to worry about the "python bindings" or the "virtual environment" to run these.

To run these examples, first change to the directory:

```
cd ~/rpi-rgb-led-matrix/examples-api-use/
```

We'll use the `demo` program to test the setup. Here is an example for a **32x32 LED matrix**. Note the program must be run with `sudo`.

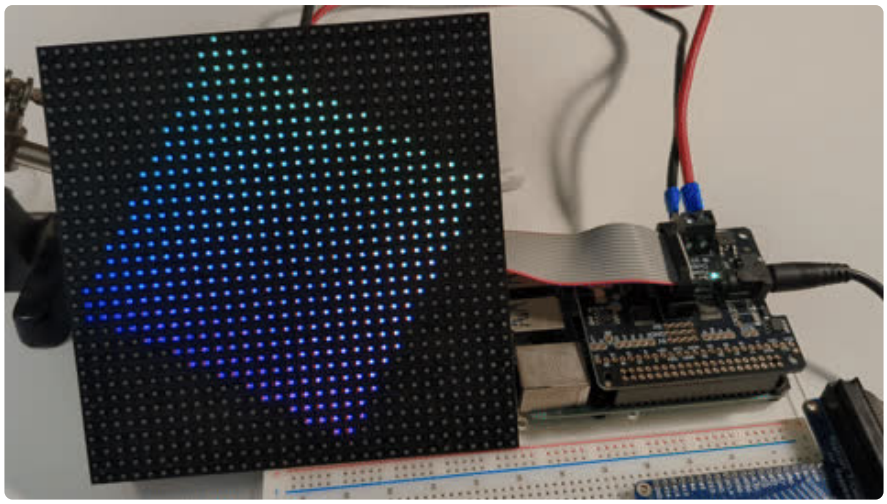
```
sudo ./demo --led-rows=32 --led-cols=32 --led-gpio-mapping=adafruit-hat -D 0
```

Change the following parameters as needed for your setup:

- `--led-row=` set this to the number of LEDs in each matrix row
- `--led-cols=` set this to the number of LEDs in each matrix column
- `--led-gpio-mapping=` set this to `adafruit-hat` or `adafruit-hat-pwm` for single matrix bonnet/HAT. For triple matrix bonnet use `regular`
- `--led-parallel=` set this to `3` if you are using the triple matrix bonnet or other active3 compatible hardware. Otherwise it can be omitted.

Running the `demo` program:


should result in a rotating square on the matrix:



The `-D` parameter picks the specific demo to run. There are several to choose from, some with additional parameters:

```
Demos, choosen with -D
    0  - some rotating square
    1  - forward scrolling an image (-m &lt;scroll-ms&gt;)
    2  - backward scrolling an image (-m &lt;scroll-ms&gt;)
    3  - test image: a square
    4  - Pulsing color
    5  - Grayscale Block
    6  - Abelian sandpile model (-m &lt;time-step-ms&gt;)
    7  - Conway's game of life (-m &lt;time-step-ms&gt;)
    8  - Langton's ant (-m &lt;time-step-ms&gt;)
    9  - Volume bars (-m &lt;time-step-ms&gt;)
    10 - Evolution of color (-m &lt;time-step-ms&gt;)
    11 - Brightness pulse generator
```

## Tuning the Demo

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

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

**For Raspberry Pi 3 use a slowdown of 1 to start (use higher values if image still flickers). For Raspberry Pi 4, use a slowdown of 4. Older Pi models might work with 0, try it.**

`--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`.

### 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 or Pi 4, you may need to dial back the matrix control speed slightly. This can be done with the `--led-slowdown-gpio=2` setting. Pi 4 may require larger values, depending on the matrix...**experiment!** 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.

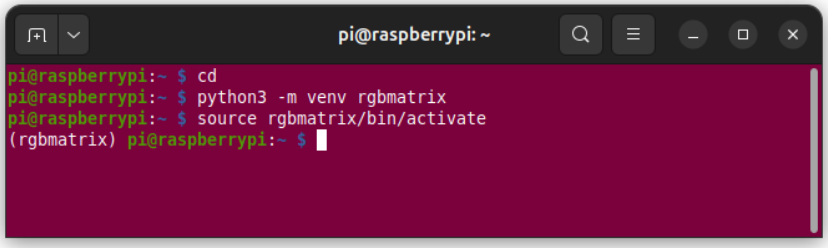
---

## Python Usage

If you manually installed things, this was likely already taken care of.

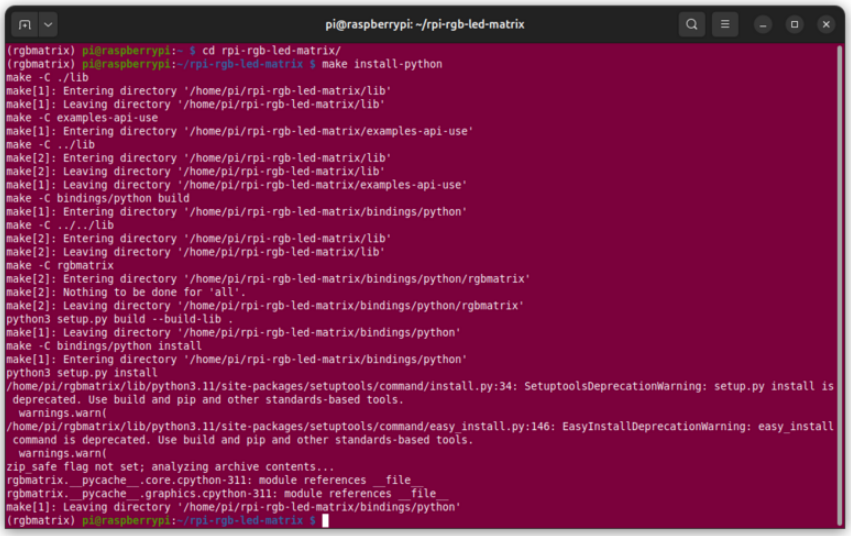
The installer script builds the Python bindings, but does not install them. We want to install things into a Python virtual environment (https://adafru.it/1a9O). So the first step is to create the venv and activate it. We'll call the venv `rgbmatrix` and locate it in the pi user's home folder:

```
cd
python3 -m venv rgbmatrix
source rgbmatrix/bin/activate
```




Next, cd back into the directory the repo was cloned into and run make with the install-python target:

```
cd rpi-rgb-led-matrix/
make install-python
```

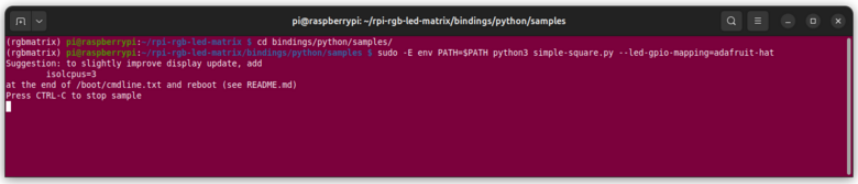



That should install the Python support into the venv.

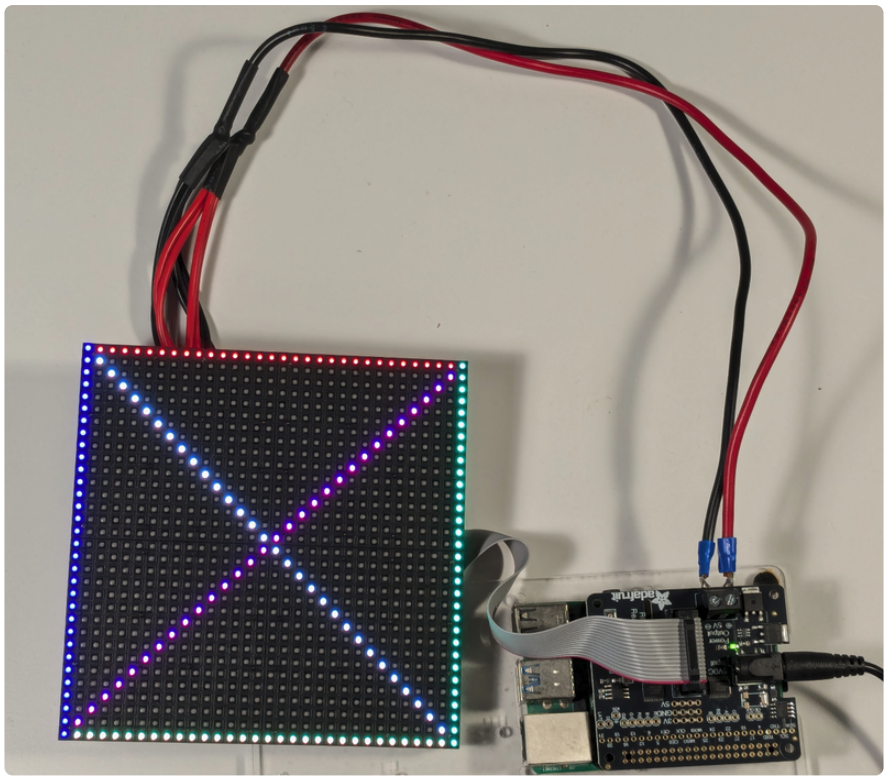
There are some Python examples found in the `bindings/python/samples` subfolder that can be used to test the setup. The examples must be run with `sudo`, which can be tricky when used within a venv. [The Python virtual environment guide has more information on how to deal with this](https://adafru.it/1a9Q) (https://adafru.it/1a9Q).

Here is an example:

```
cd bindings/python/samples/
sudo -E env PATH=$PATH python3 simple-square.py --led-rows=32 --led-cols=32 --led-
gpio-mapping=adafruit-hat
```




That should result in output on the RGB matrix that looks like this:



Parameters can be set via the command line. Change the following parameters as needed for your setup:

- `--led-row=` set this to the number of LEDs in each matrix row (default=32)
- `--led-cols=` set this to the number of LEDs in each matrix column (default=32)
- `--led-gpio-mapping=` set this to `adafruit-hat` or `adafruit-hat-pwm` (default=regular)

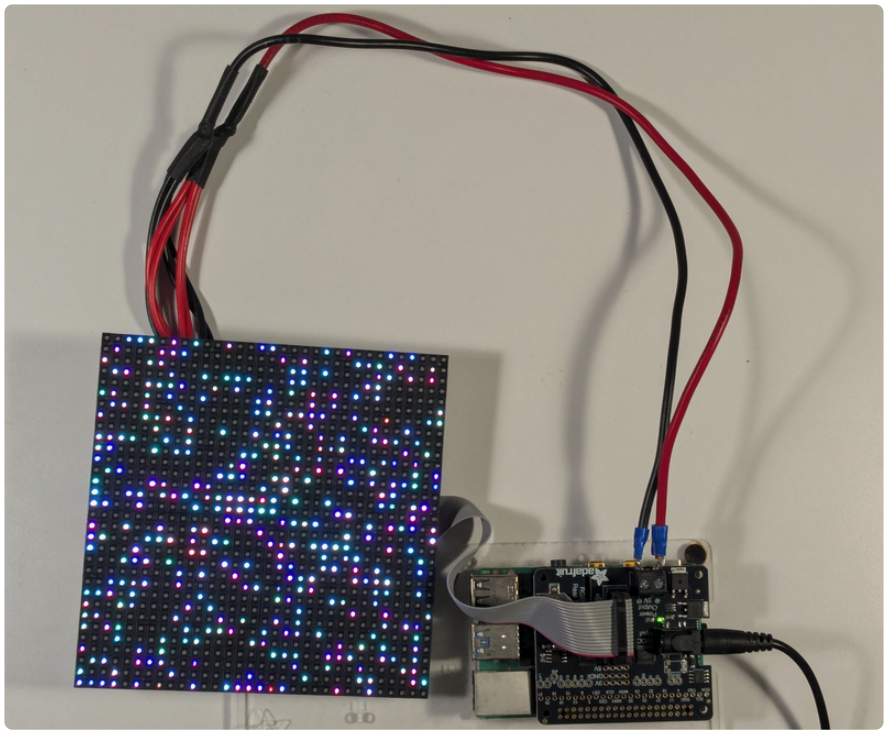
## Another Basic Example

The examples in the library repo have a sort of object-orientated design which may make things look overly complicated. Additionally, there is a lot of code dealing with parsing the command line options. All of the command line parameters can be configured directly in code.

Here is another example to show more simply how the RGB matrix can be used in Python:

```
import time
from random import randrange
from rgbmatrix import RGBMatrix, RGBMatrixOptions

options = RGBMatrixOptions()

options.hardware_mapping = 'adafruit-hat'
options.rows = 32
options.colrs = 32

matrix = RGBMatrix(options = options)
```

```
while True:
    matrix.Clear()
    for _ in range(1000):
        r = randrange(256)
        g = randrange(256)
        b = randrange(256)
        x = randrange(32)
        y = randrange(32)

        matrix.SetPixel(x, y, r, g, b)

        time.sleep(0.01)
```



The general idea is to create an instance of the `RGBMatrixOptions` class and then set things as desired. Then create an instance of the `RGBMatrix` class and pass in the options.

The only documentation other than the examples is the source code itself, as linked from the repo readme (https://adafru.it/BhW). Here is a terse summary of the available functions and properties.

The `RGBMatrixOptions` class has the following properties:

```
brightness
chain_length
cols
daemon
disable_hardware_pulsing
drop_priv_group
drop_priv_user
drop_privileges
gpio_slowdown
hardware_mapping
inverse_colors
```

```
led_rgb_sequence
limit_refresh_rate_hz
multiplexing
panel_type
parallel
pixel_mapper_config
pwm_bits
pwm_dither_bits
pwm_lsb_nanoseconds
row_address_type
rows
scan_mode
show_refresh_rate
```

The `RGBMatrix` class has the following functions and properties:

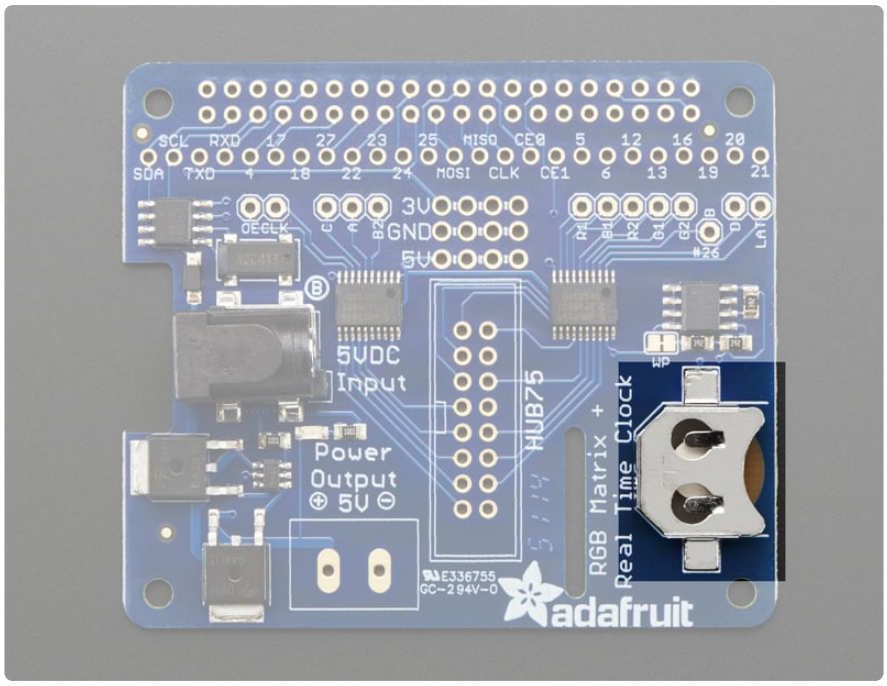
```
Clear()
CreateFrameCanvas()
Fill()
SetImage()
SetPixel()
SetPixelsPillow()
SwapOnVSync()
brightness
height
luminanceCorrect
pwmBits
width
```

---

# Using the RTC

We had a little space and thought a real time clock would be a nice pairing for this HAT so we tossed on a DS1307 real time clock (RTC). This clock uses a 32.768KHz crystal and backup battery to let the HAT & Pi keep track of time even when power is lost and there's no network access. This makes it great for time displays!

A 12mm 3V Lithium Coin Cell (CR1220) is REQUIRED to use the RTC! It will not work without one! **(http://adafru.it/380)**




The `rgb-matrix.sh` script already installed the necessary software to use the realtime clock...but you'll need to set the initial time once. This is explained in the **"Sync time from Pi to RTC"** section of this DS1307 tutorial (https://adafru.it/IF1) (just that one section...you can ignore the rest).

---

# HELP!

## The matrix looks like it's trying to work, but the image is all flickery.

Try using the `--led-slowdown-gpio=X` command-line setting, where "X" ranges from 0 (for the very earliest Raspberry Pi models) up to 4 (for the Raspberry Pi 4 and 5). Experiment to find the **lowest value** that provides a **stable image** on your system.

## 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 prior note about dialing back the GPIO speed.

## The matrix flashes on and off when in use

If you're interfacing to any **1-wire** devices, and if you've enabled 1-wire via raspi-config, you'll need to use something other than the default pin 4. Pins 19 or 25 make good choices. Look for the line in **/boot/config.txt** where 1-wire is enabled and tell it which pin to use:

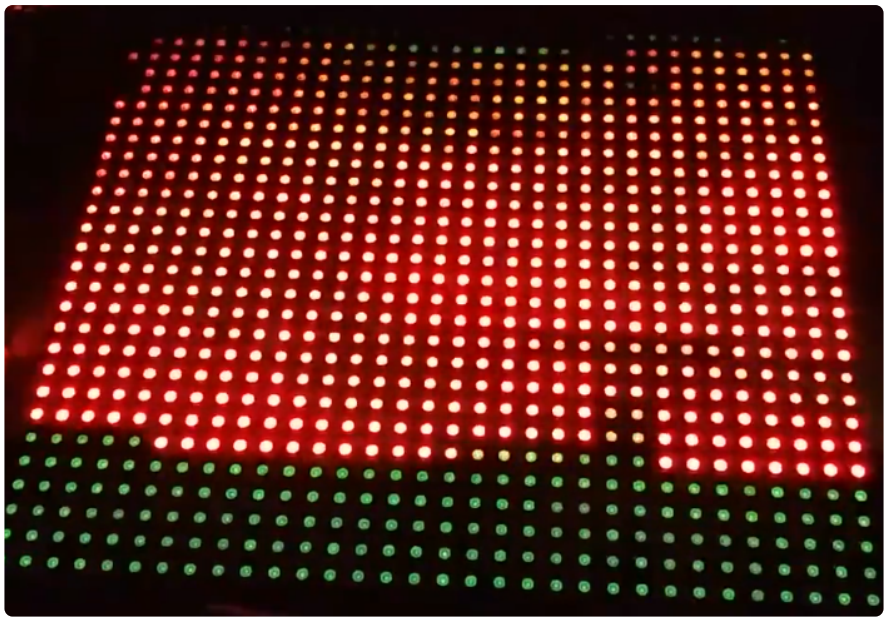
```
dtoverlay=w1-gpio gpiopin=19
```

## Tried all these, it still flickers.

Large LED matrices and newer model Raspberry Pis need more power than the early days. If you're trying to run everything from a 5V 2 Amp power supply, you probably need to step up to 4A or better. You can also try powering the Raspberry Pi from a USB power supply and the matrix (via the HAT or Bonnet) with a DC supply.

## My display has dim red LEDs or has odd output

Check that you have plugged in a good 5V 2A+ (4A+ is best) power supply into the Bonnet/HAT. **The Raspberry Pi's 5V power supply cannot power a matrix, you need a separate high current supply as well!**



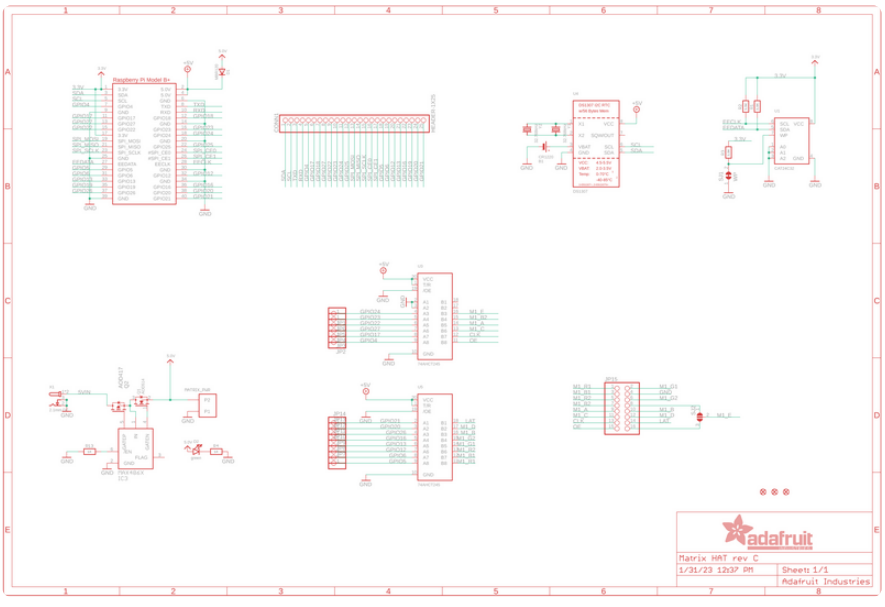
# Downloads

## Datasheets

- DS1307 Real Time Clock (https://adafru.it/em5)
- 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/qHc)

# Schematic

Click to embiggen



# Fabrication Print

Here's the fabrication print with dimensions in inches. This HAT is compatible with the Raspberry Pi mechanical HAT specification!