DotStar Pi Painter

Ultimate light-painting rig!

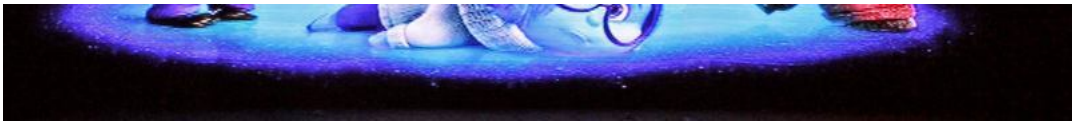
- Overview
- Raspberry Pi Setup
- Assembly Part 1
- Assembly Part 2
- Assembly Part 3
- Photograph!
- Image Prep and Fine-Tuning
- Single Page
- Download PDF

Contributors

Phillip Burgess
Feedback? Corrections?
RASPBERRY PI LEDS / LED STRIPS

Assembly Part 3

by Phillip Burgess



Home stretch! Nearly there! At this point, you should have LEDs mounted on a support bar, the Raspberry Pi (with HAT) in an enclosure, and have the strandtest code working.

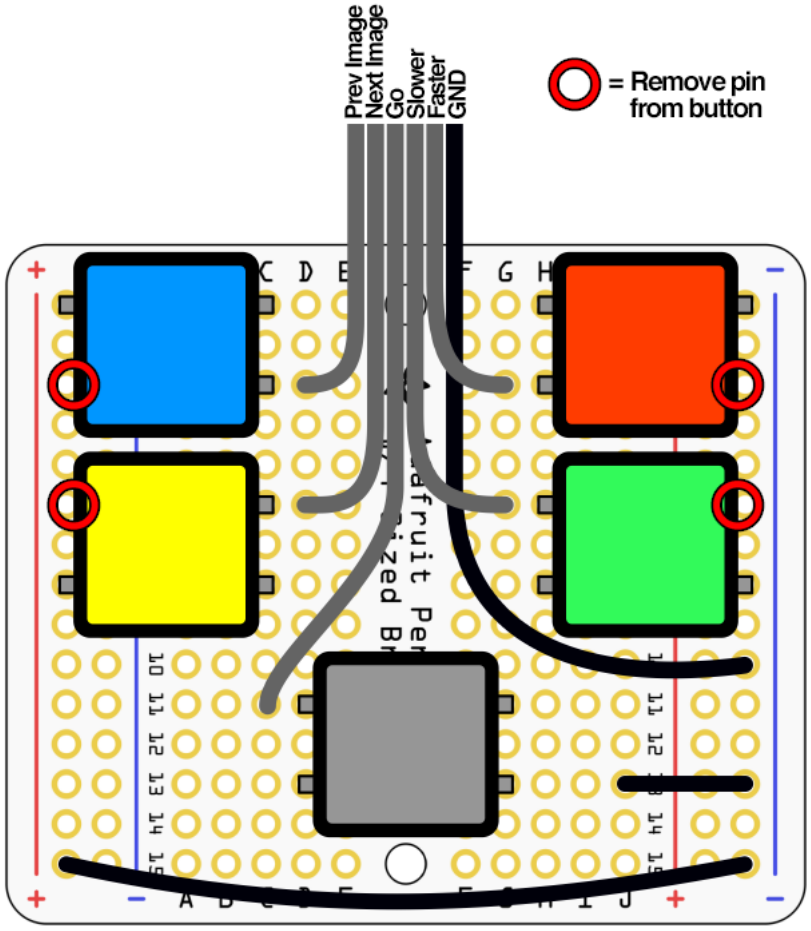
Next we'll add painting controls, mount the battery to the frame and optionally add a light diffuser.

Control Buttons

Everyone loves buttons! We'll add a few to trigger the light painter, adjust speed and select images.

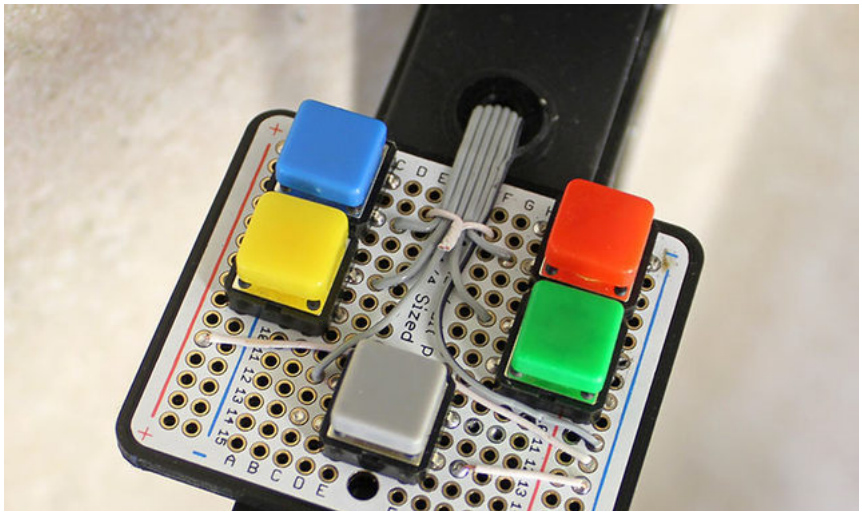
The controls should be near the handle. Since the Pi is kept close to the strip input, you'll need six long wires or some ribbon cable to bridge the distance.

I used five tactile buttons soldered on a 1/4-size Perma Proto breadboard. Any sort of momentary buttons will do, if you already have some around.



Disregard the "+" label and join both outermost rails to ground.

On the four buttons in contact with the ground rails, it was necessary to **clip away one pin using flush cutters**. This should be the pin **straight across** from each button's corresponding ribbon cable wire.

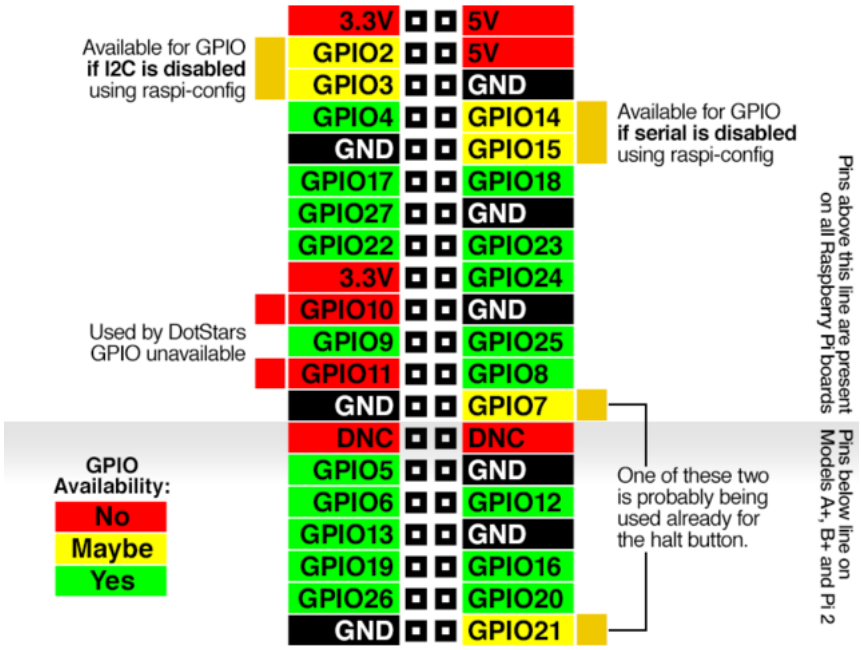


A little double-stick foam tape is sufficient to hold the controls in place, but I went nuts with 3D printing and made this little press-fit enclosure that holds the Perma Proto and buttons, with an escape slot for a ribbon cable at the top.

- 
- 
- 

The ribbon cable (or separate wires, if that's what you have) then snake their way back to the Raspberry Pi, where they're soldered to points on the Pi HAT (or add a socket connector).

A sixth button (not shown here) can go on the Pi HAT for the halt function. I happened to use a smaller tactile button for that (shown on the prior page), but it's all good.

- 

For posterity, here's the GPIO pin map again. Any of the green pins is fair game for each of the five buttons…

…but here are the pin assignments normally used in the software:

Copy Code

```
pin_go     = 22
pin_next   = 17
pin_prev   =  4
pin_faster = 23
pin_slower = 24
```

If you follow this pinout, then your painter will continue to work after any updates to our code; you won't have to edit that part each time.

Battery Mount

To hold the USB battery bank, I made this 3D-printed bracket and affixed it to the frame with E6000 glue. Battery then clips into place.

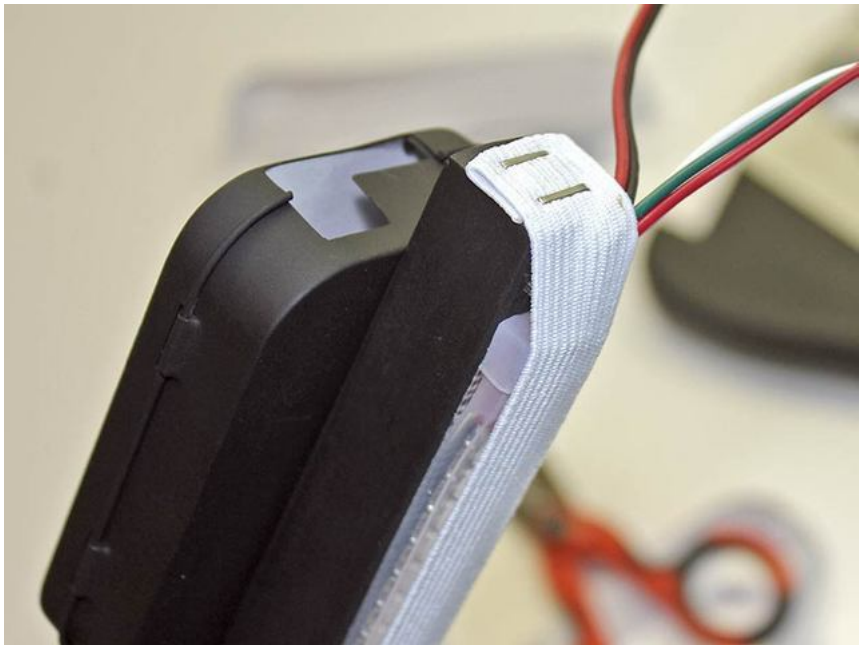
No 3D printer? No problem. Double-stick foam tape will work as well.

Pressing the button on the battery bank switches it on, but also turns on an annoying blue LED that will ruin photographs. **Pressing the button a second time turns off the LED, but continues to deliver USB power.** To power down the Pi and LED strip, physically unplug the USB cables, which will

also shut off the battery bank.

Diffuser

Slightly diffusing the light from the LEDs makes them photograph better. You get a wash of light over a small area rather than a single focused dot.



In the NeoPixel Painter guide we used **3/4" elastic** as a diffuser. Still the recommended approach!

In my overly-complex DotStar Pi Painter build, the same channel that holds the LED strip has a second set of grooves to hold elastic, pulled taut and secured at the ends with more 3D-printed bits.

First Light

Now lets give it a test run!

Plug the LED strip and the Raspberry Pi into the USB battery bank and switch it on. While that boots, on your “main” computer, dig up a **USB flash drive**, format it as a Windows FAT32 filesystem, and drop a few **JPEG, PNG or GIF** images on there. These should go at the root level of the drive, not inside any folders.

Log in to the Raspberry Pi (either ssh over wireless, or plug in display and keyboard) and run the light painting script manually:

Copy Code

```
cd DotStarPiPainter
sudo python DotStarPiPainter.py
```

Nothing will happen at first. Plug in the USB flash drive and you should see a message as it loads and does some processing on the first image. The LED strip will also give a simple indication as its working…red while loading, yellow while processing, green indicating success (ready). The position along the strip shows which image number is being loaded.

If that works, press the “go” button.

You should see the LEDs glimmer for a few seconds, though it won’t make any sense to the naked eye.

Try the faster & slower buttons. A blue light should move up and down the strip indicating the duration of the paint time; about 0.1 second at one end of the strip, 10 seconds at the opposite end. The default time is 2 seconds. Try longer and shorter durations and press the “go” button again.

Try the next & previous image buttons. If there’s multiple images on the USB drive, it’ll show the red-yellow-green sequence as a new image is loaded. Press the “go” button again.

No lights! No go!

First, confirm that the USB drive is detected and contains valid image files; **you'll see the DotStarPiPainter.py script printing messages as it works.**

If images are loading but nothing's displayed, refer to the troubleshooting steps on the Assembly Part 1 page.

Troubleshooting

I got some lights for a moment, but now it's just stuck there!

This can happen if the LEDs draw a lot of current. Voltage from the battery bank dips and the Raspberry Pi locks up. We can fix this!

Reboot the system (unplug, re-plug USB), wait for it to boot, log in and then edit the DotStarPiPainter.py script.

Copy Code

```
1. cd DotStarPiPainter
2. nano DotStarPiPainter.py
```

Look for the following line of code (around line 74, just before "INITIALIZATION"):

Copy Code

```
1. power_settings = (1450, 1550)
```

Those two numbers are the average and peak current (very approximate-ish) to deliver to the LEDs. Although the battery bank *can* provide more *current* than this, the *voltage* drops when it does so. We need to dial it back a bit. Try changing this to (1000, 1000), save the changes and re-run the script.

If it's successful now, take note of the value(s) last used, then try something a little higher…like (1200, 1200). If it works, you can continue dialing it up in smaller increments. If it locks up again, reboot and dial down.

There is a small but nonzero chance of the SD card being corrupted during these unplanned reboots, requiring a reformat and reinstall. Once you get it everything dialed in and fully set up, make a backup image of the card just in case!

Another option, if you've got a second battery bank around: dedicate the larger/higher-current pack 100% to the LEDs and the second one fully to the Pi. The DotStar LEDs work fine with some voltage drop, colors should remain true even at 4V, whereas the Pi will start to have trouble. So if you give the LEDs their own big battery, you can push the current up quite a bit (e.g. (2100, 2600) or so)…the Raspberry Pi, on its own dedicated battery supply, will be totally unaffected by the heavy current draw. The Pi is perfectly happy on a smaller phone-charging battery bank (e.g. 500 mA) if that's what you have onhand.

Set Up Auto-Start

If everything looks good, last step is just to run the DotStarPiPainter.py script automatically when the system boots. If it's currently running, press Control+C to stop the program. Then enter:

Copy Code

```
1. sudo nano /etc/rc.local
```

Just before the final "exit 0" line (and after the "gpio-halt" line, if you've installed that), insert the following line:

Copy Code

```
1. python /home/pi/DotStarPiPainter/DotStarPiPainter.py
```

Now walk through a full power cycle to confirm everything's set:

- Press the halt button.
- Wait at least 15 seconds.
- Unplug both USB cables.

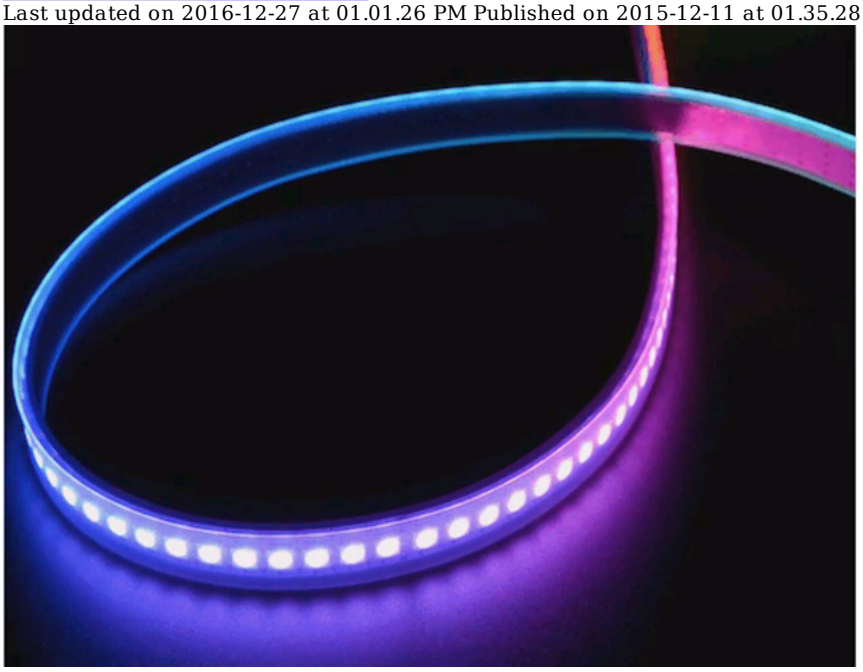
System is now fully powered down. The USB battery bank will sleep automatically when there's no load. Now let's boot:

- Plug in both USB cables; LEDs to the high-current port, Pi to the other port.
- Wait 30-60 seconds for system to boot.
- Plug in a USB drive containing images. You should see the loading sequence on the LEDs.
- Press the "Go" button. LEDs should activate.

Does it work? Grand! You can remove the WiFi dongle now; it's not needed during normal use.

If it *doesn't* work (but did pass the "first light" test), it's probably a typo in the rc.local file.

ASSEMBLY PART 2 PHOTOGRAPH!
Last updated on 2016-12-27 at 01.01.26 PM Published on 2015-12-11 at 01.35.28 PM

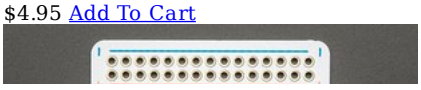


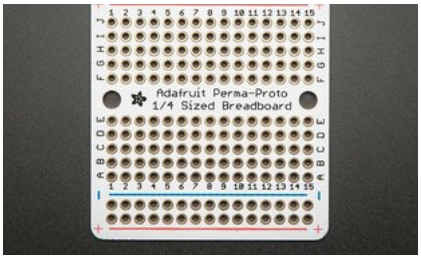
Adafruit DotStar Digital LED Strip - White 144 LED/m - One Meter
$74.95 Add To Cart

Raspberry Pi 2 - Model B - ARMv7 with 1G RAM
$39.95 Add To Cart

Raspberry Pi Model A+ 512MB RAM
$24.95 Add To Cart

Adafruit Perma-Proto HAT for Pi Mini Kit - No EEPROM
$4.95 Add To Cart





Adafruit Perma-Proto Quarter-sized Breadboard PCB - Single
$2.95 Add To Cart

74AHCT125 - Quad Level-Shifter (3V to 5V)
$1.50 Add To Cart


Colorful Square Tactile Button Switch Assortment - 15 pack
$5.95 Add To Cart


USB Battery Pack for Raspberry Pi - 10000mAh - 2 x 5V outputs
$39.95 Add To Cart


USB DIY Connector Shell - Type A Male Plug
$0.95 Add To Cart
ADD ALL TO CART

RELATED GUIDES

Install bluez on the Raspberry Pi

How to compile and install bluez, the Linux Bluetooth classic & low energy system, on the Raspberry Pi.
by Tony DiCola

Learn how to download and install the latest version of the bluez Bluetooth library on the Raspberry Pi. Start using your Pi with Bluetooth classic & low energy in no time!

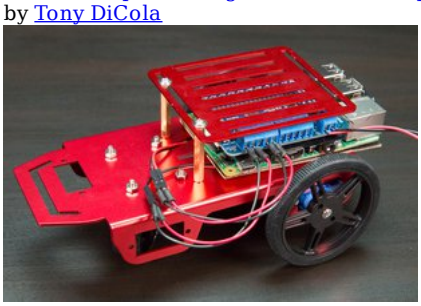
Raspberry Pi Analog to Digital Converters

How to read analog signals from Python with an analog to digital converter and Raspberry Pi.
by Tony DiCola

Learn how to connect a MCP3008 or ADS1x15 analog to digital converter to a Raspberry Pi and use it to read analog signals from Python code.

Simple Raspberry Pi Robot

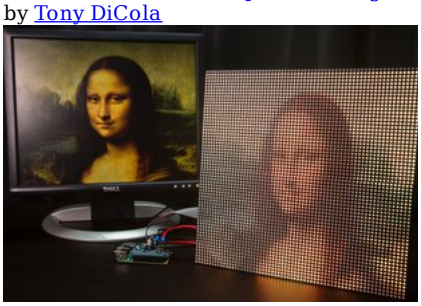
Build a simple moving robot with a Raspberry Pi and Adafruit's excellent robot platform!
by Tony DiCola



Use a Raspberry Pi and Adafruit Motor HAT to control two DC motors that move a simple robot around using Python code. This is a great base for starting with a Pi robot project!

Raspberry Pi LED Matrix Display

Show the Pi's video output on a large RGB LED matrix display!
by Tony DiCola



Learn how to make a chain of RGB LED matrices into a large display that shows the video output of a Raspberry Pi. Great for playing games, movies, and more on a big, bright, beautiful display!
×

OUT OF STOCK NOTIFICATION

YOUR NAME
YOUR EMAIL
NOTIFY ME


VISA
MasterCard
AMERICAN EXPRESS
PayPal
amazonpayments
DISCOVER