Light Painting with Raspberry Pi

Created by Phillip Burgess

https://learn.adafruit.com/light-painting-with-raspberry-pi

Last updated on 2025-01-22 12:14:04 PM EST

Table of Contents

Overview 3

Hardware 4

Software 7

Motion Rig 10

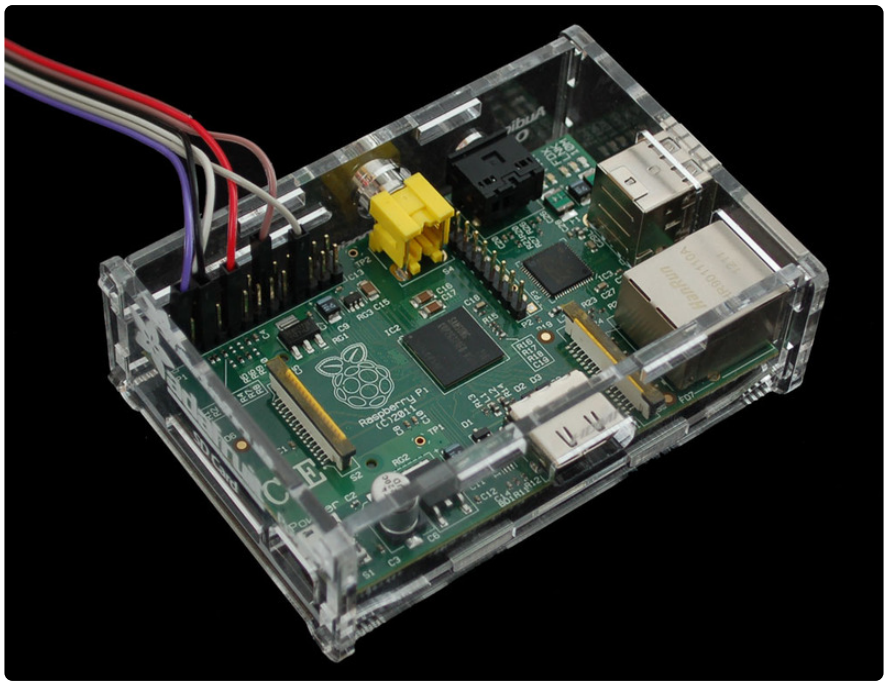
Results 11

Overview

Light painting is an artistic photographic technique combining long exposure times with lights in motion. Traditionally these images have been hand-painted with a penlight...but more recently, affordable microcontrollers and addressable RGB LEDs have brought a new high-tech twist to the idea.

A few such projects have been featured on tech blogs, most notably the (https://adafru.it/aND)LightScythe (https://adafru.it/aND) by Gavin "The Mechatronics Guy," (https://adafru.it/aND) which uses an Arduino and a two meter bar of LEDs. The resulting photos, with their intense colors and mid-air suspended images, have been super popular, and rightfully so.

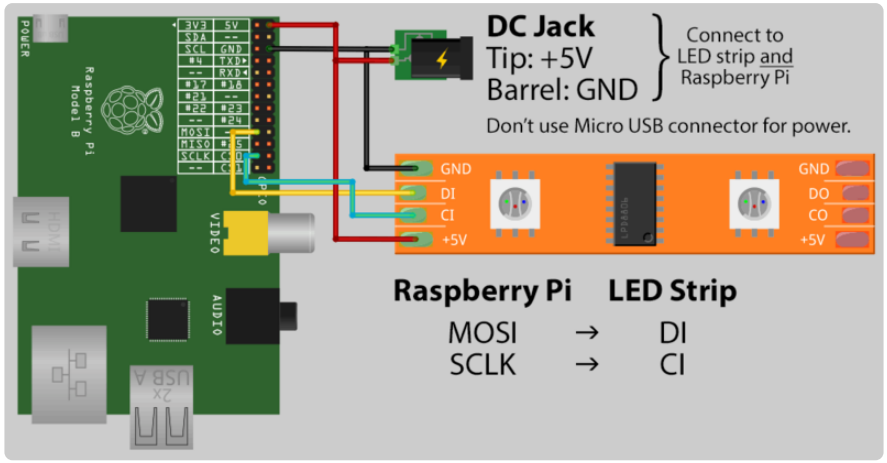
Large, colorful images require a lot of memory...and that's one area where we'll need more memory than an Arduino can provide. We had a hunch that the Raspberry Pi could make this process easier. But even we weren't prepared for what a cakewalk this would be...



Hardware

Interfacing Adafruit's Digital Addressable RGB LED (https://adafru.it/aHG) strip (aka "LPD8806 strip") to the Raspberry Pi is super simple, requiring just a few connections between the board, strip and a DC power jack (http://adafru.it/368).

The board's MOSI pin connects to the DI pin on the LED strip, and SCLK connects to the CI pin.

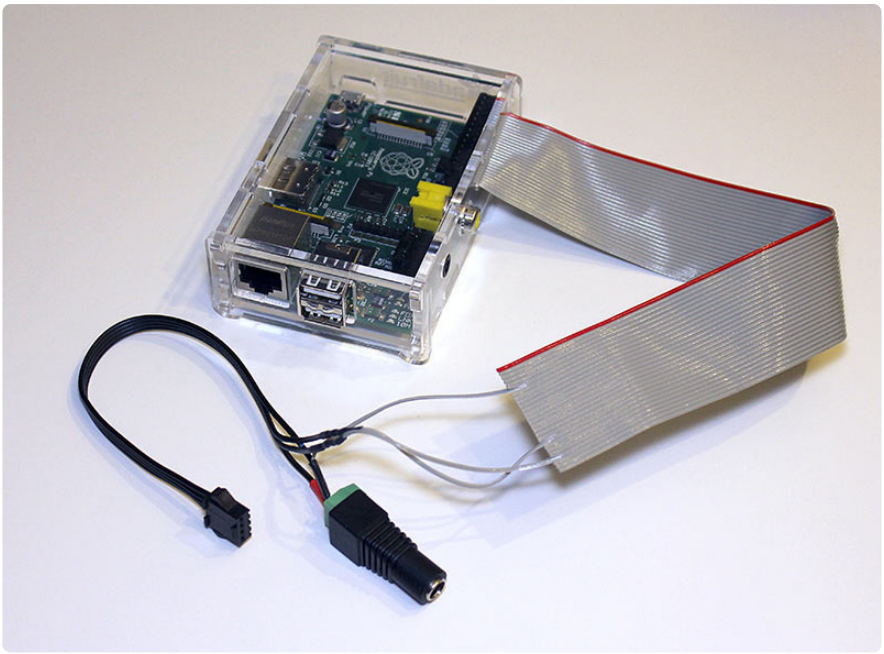


Instead of supplying power to the Raspberry Pi's Micro USB connector, a 5 Volt DC power supply is required because the LED strip draws significantly more current than the USB input can provide. A 2 Amp power supply (http://adafru.it/276) is sufficient for a 1 meter LED strip, while our larger 10 Amp supply (http://adafru.it/658) can power up to 5 meters of LED strip (plus the Raspberry Pi board, in both situations).

+5V and ground from the power supply connect to the 5V and GND pins on both the LED strip and the Raspberry Pi GPIO header.

In the above diagram, we're directly connecting **3.3V** logic output from the Raspberry Pi to the **5V** logic input of the addressable LED strip. Strictly speaking, this is not Good and Proper. Wildcards like actual power supply voltage (they're never precisely 5.0000V) or even temperature may contribute to whether this functions reliably. It did very well here, but your mileage may vary. If your LEDs almost work but are glitchy, it's time for a logic level shifter, explained in this guide (https://adafru.it/FXc) (which is about NeoPixels, but the principle is the same — just that we've got two wires to level-shift here, rather than NeoPixels' one).

An initial prototype was assembled using a Pi Cobbler (http://adafru.it/914) breakout kit. Because the finished project would be moving around a lot, and because breadboards aren't the most robust thing, a 26-pin IDC cable was sacrificed for science to create a purpose-built cable between the Raspberry Pi GPIO header, LED strip and power supply. This is much more resilient to vibration and careless fingers.



To make connections easy to make/break for setup and take down, we also used two JST 4-pin inline cables (plug (http://adafru.it/578)and receptacle (http://adafru.it/579)) and to attach and detach the LED strip. The connectors are polarized so they can't be plugged backwards!

Software

The Digital LED strip requires very fast data signal and you can't just use the GPIO libraries to do it. You must have hardware SPI installed, this is not included in the default "Wheezy" distribution and requires a firmware update, so please use Occidentalis!

On the software front, we started with “Occidentalis,” the Adafruit Raspberry Pi Educational Linux Distro (https://adafru.it/aNv). This has a few features baked in that make this project a lot simpler, including hardware SPI support...it’s super fast and easy, with no bitbang kludges required. Alternatively you can install SPI support yourself (https://adafru.it/BJZ)

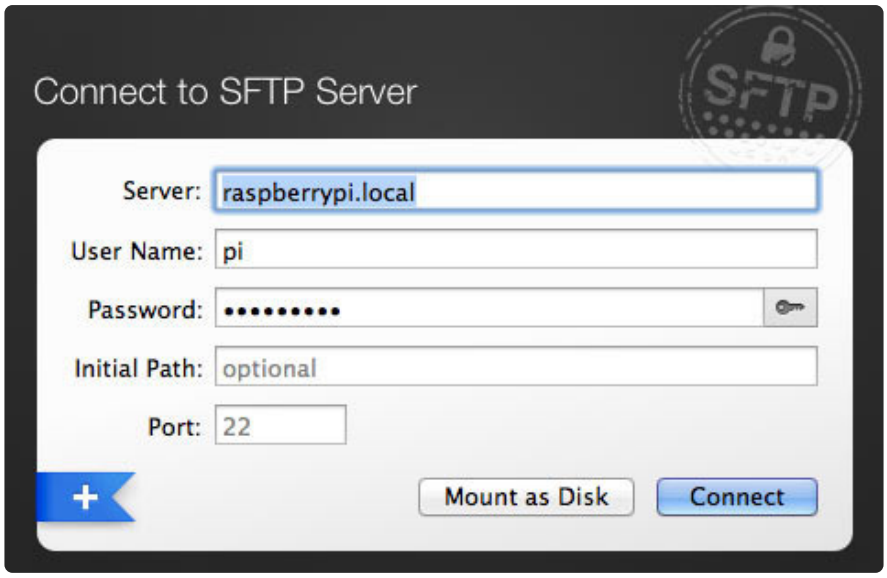
Our light painting script relies on the Python Image module, which is not installed by default. With your Raspberry Pi connected to the internet, log in and type:

sudo apt-get install python-imaging

Occidentalis also has sshd built in, making it easy to get images from a desktop machine to the Raspberry Pi. For example, in a Terminal window on my Mac I could type:

scp image.png pi@raspberrypi.local:

Or use a secure FTP client:



Here's the complete Python script for the light painting project. Even with lots of comments, it's not a large program at all:

```
#!/usr/bin/python

# Light painting / POV demo for Raspberry Pi using
# Adafruit Digital Addressable RGB LED flex strip.
# ----&gt; http://adafruit.com/products/306

import RPi.GPIO as GPIO, Image, time

# Configurable values
filename  = "hello.png"
dev       = "/dev/spidev0.0"

# Open SPI device, load image in RGB format and get dimensions:
spidev    = file(dev, "wb")
print "Loading..."
img       = Image.open(filename).convert("RGB")
pixels    = img.load()
width     = img.size[0]
height    = img.size[1]
print "%dx%d pixels" % img.size
# To do: add resize here if image is not desired height

# Calculate gamma correction table.  This includes
# LPD8806-specific conversion (7-bit color w/high bit set).
gamma = bytearray(256)
for i in range(256):
    gamma[i] = 0x80 | int(pow(float(i) / 255.0, 2.5) * 127.0 + 0.5)

# Create list of bytearrays, one for each column of image.
# R, G, B byte per pixel, plus extra '0' byte at end for latch.
print "Allocating..."
column = [0 for x in range(width)]
for x in range(width):
    column[x] = bytearray(height * 3 + 1)

# Convert 8-bit RGB image into column-wise GRB bytearray list.
print "Converting..."
for x in range(width):
    for y in range(height):
        value = pixels[x, y]
        y3 = y * 3
        column[x][y3]     = gamma[value[1]]
```

```
        column[x][y3 + 1] = gamma[value[0]]
        column[x][y3 + 2] = gamma[value[2]]


# Then it's a trivial matter of writing each column to the SPI port.
print "Displaying..."
while True:
    for x in range(width):
                spidev.write(column[x])
                spidev.flush()
        time.sleep(0.001)
    time.sleep(0.5)
```

The script needs to be run as “root” because it accesses the GPIO hardware, i.e.:

sudo python lightpaint.py

After opening the SPI device (for communicating with the LED strip), the script then loads an image using the Python Image module, converting it to RGB format if necessary. As an interpreted language, Python isn’t always the quickest thing…rather than repeatedly process each row or column of an image on the fly, the entire image is pre-processed once from the native RGB format into the hardware-specific format required of the LED strip, held in a list of bytearrays. We can then quickly dump each of these arrays directly to the SPI port without any further work. Decoding and holding all this intermediate data would be inconceivable on Arduino!

The software is smart enough that it will use the height of the image as the number of pixels to address. So if you have 64 pixels, make sure your images are 64 pixels tall, etc!

This runs in an infinite loop. Hit **CTRL+C** or locate and kill the process to make it stop.

The software runs, but I’m not seeing any LEDs light up

Common causes, in decreasing order of likelihood:

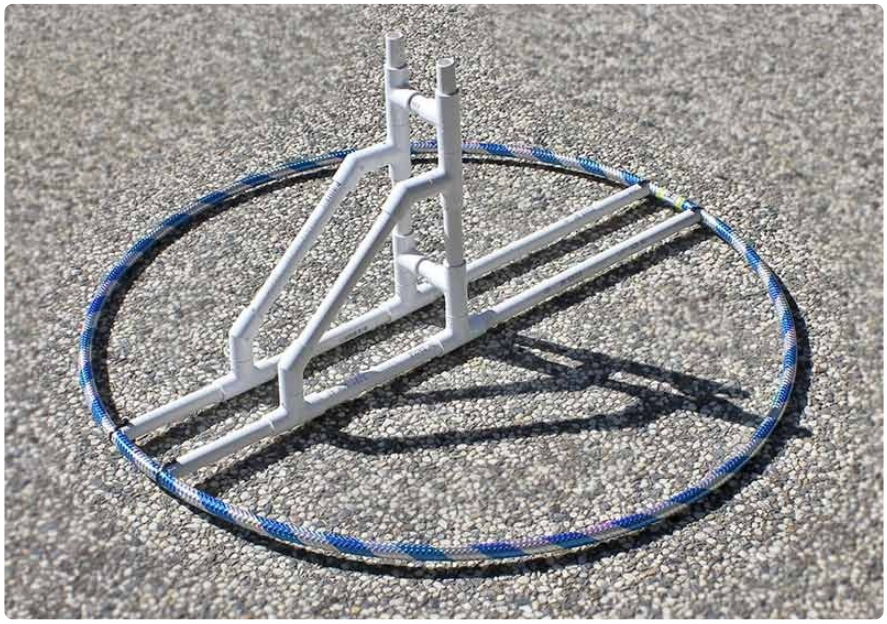
- The clock and data lines might be switched. Try the other way, and if no difference, put them back and double-check against the wiring diagram and the pin labels on the strip.
- Might’ve accidentally connected to the output end of the strip rather than the input. Make sure you soldered to “CI” and “DI,” not “CO” and “DO.” Some strips may have arrows showing the data direction from in to out.
- Did you remember the ground wire between the LED strip and Raspberry Pi?

- You might need a logic-level shifter, explained a bit on the "Hardware" page.

Motion Rig

Handheld, straight LED bars have been done. With the code so quickly out of the way, we wanted to take this to the next level. First, the light bar would be replaced with a circle, in order to give the finished pictures an interesting three-dimensional quality. Second, it would be attached to a bicycle to provide smooth motion and to cover much longer distances. Riding this through the darkness during a long-exposure photograph should then create an extruded tube in 3D space. The bicycle "disappears" in the image because it doesn't sit still long enough for the camera to expose.

A contraption was quickly assembled from PVC pipe and a hula hoop, then spray painted matte black to be stealthy for photos. This rig would attach with zip ties to the rack over the back wheel of the bike.

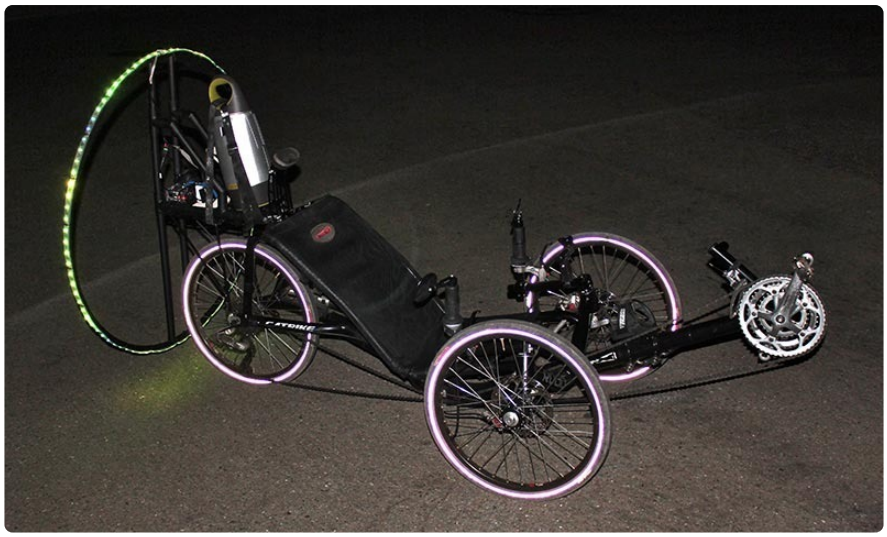


For power to both the LEDs and computer, a camping battery/inverter was used. The absurdity of this is not lost on us — converting 12V DC to 110V AC back down to 5V DC — but time was of the essence and the portable power unit was already on-hand (and when installed on the bike it looks like a Mr. Fusion...cool!). For something less hack-ish, a DC-to-DC converter would make a lot of sense.

A strip of 96 LEDs was used to go around the hoop

Generous application of zip ties and tape complete the mad science project look. This won't be surviving a trip to Mars any time soon.

Yes, that's my bike. Er, trike. What? Why are you looking at me funny?



Results

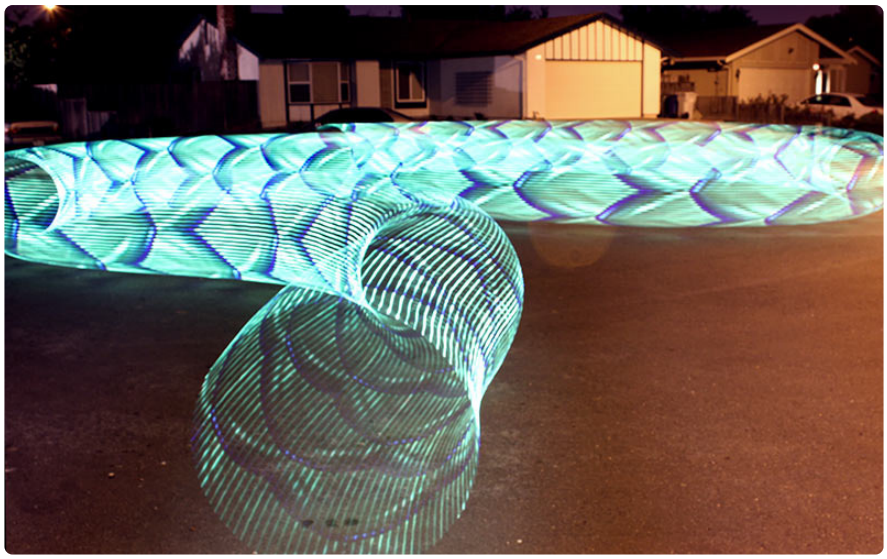
These first images were cobbled together pretty quickly, and we didn't go out of our way for a good, dark location. With some planning and refinement the results can potentially be much nicer than this.

These are all 10- to 15-second exposures.

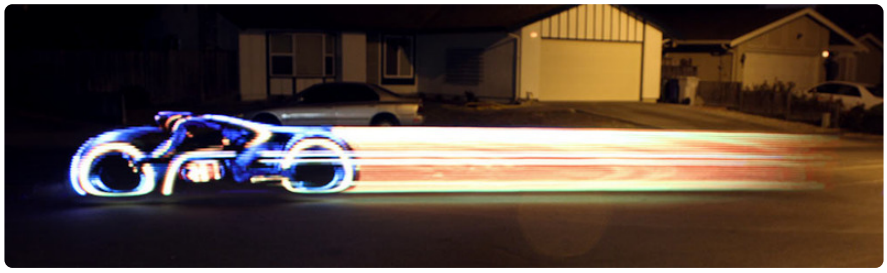
The ring of fire:

Cola wars:

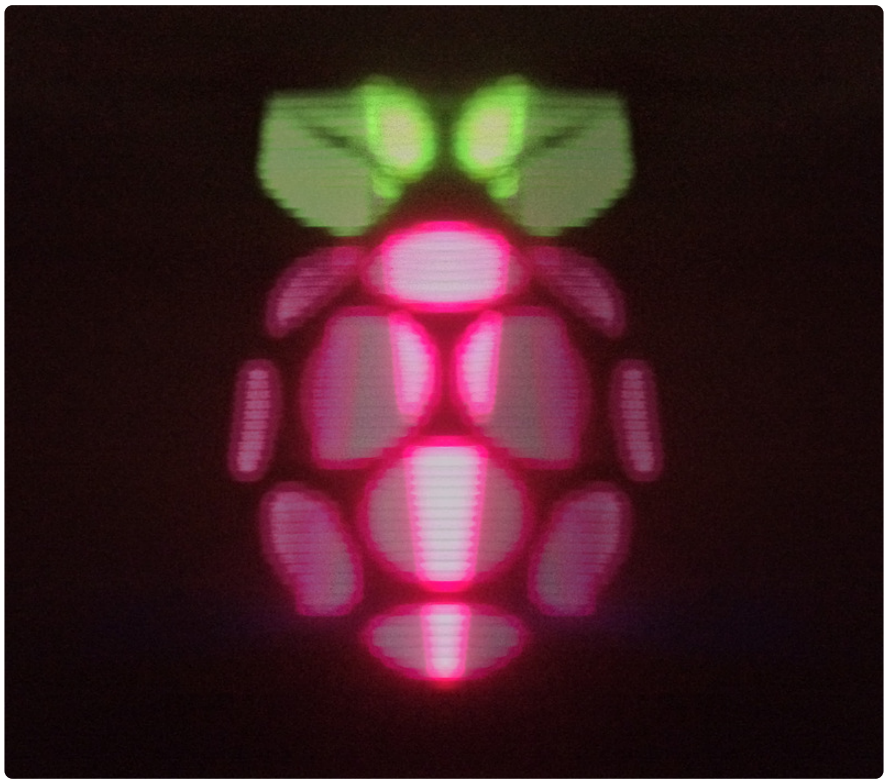
Big snake. Ostensibly a python:



Tron Legacy light cycle:



Total project time, starting from zero Python experience to having a working demo, photographs and a tutorial, was about two days. Great things are afoot!



Update: Here's a quick test Ladyada did in Adafruit warehouse last night, shot with a phone!