adafruit learning system

RGB & HSV NeoPixel Dialer

Created by Liz Clark

Last updated on 2018-08-22 04:06:04 PM UTC

Guide Contents

Guide Contents 2
Introduction 3
Materials 3
Circuit 5
RGB CircuitPython Code Using the NeoPixel Library 6
HSV CircuitPython Code Using the FancyLED Library 9
3D Printing 12
Case Design 12
Printing 12
LED Diffusers 13
Assembly 15
Finishing Touches 17

Introduction

Have you ever wanted a certain color for a NeoPixel project and had trouble figuring out which RGB or HSV values to enter? You can use web colors to try and get specific colors but you never know how it will look in real life until you load up your code and see it in action. This project allows you to use potentiometers to adjust the red, green and blue values individually in real time to fine tune NeoPixel colors and to see exactly how RGB and HSV values from the code work in practice.

Materials

1 x Feather M0 Express
Microcontroller to run CircuitPython
ADD TO CART

1 x Standard LCD 16x2 + extras - white on blue
LCD screen to display data
ADD TO CART

1 x NeoPixel Stick - 8 x 5050 RGB LED with Integrated Drivers
NeoPixels
ADD TO CART

4 x Panel Mount 10K Log Potentiometer
Potentiometers - 3 for RGB and 1 for the LCD brightness
ADD TO CART

1 x Super Bright Red 5mm LED
5mm Red LED for pot
ADD TO CART

1 x Super Bright Blue 5mm LED
5mm Blue LED for pot
ADD TO CART

1 x Super Bright Green 5mm LED

5mm Green LED for pot

ADD TO CART

1 x 5mm Plastic Flat LED Holder - Pack of 5

LED holders to mount individual 5mm LEDs

ADD TO CART

2 x Potentiometer Knob - Soft Touch T18 - White

White Pot knob - 1 will be painted green

ADD TO CART

1 x Potentiometer Knob - Soft Touch T18 - Red

Red pot knob

ADD TO CART

1 x Potentiometer Knob - Soft Touch T18 - Blue

Blue pot knob

ADD TO CART

1 x Adafruit Perma-Proto Half-sized Breadboard PCB - Single

Board for soldering the circuit

ADD TO CART

1 x Hook-up Wire Spool Set - 22AWG Solid Core

Wire for the circuit

ADD TO CART

1 x M2.5 Screws

Screws for mounting the Feather and Perma Proto in the bottom of the case

ADD TO CART

1 x PLA Filament for 3D Printers - 1.75mm Diameter - 1KG - Silver

Silver filament for the case

ADD TO CART

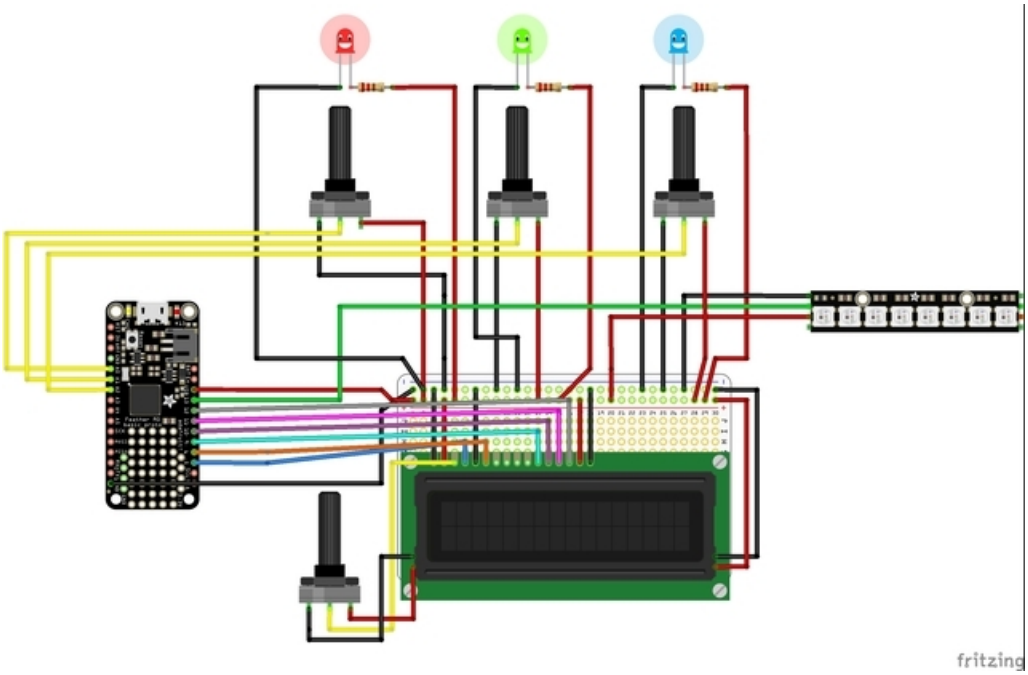
1 x PLA Filament for 3D Printers - 1.75mm Diameter - White - 1KG

White filament for the 5mm LED diffusers

ADD TO CART

Circuit

The Feather M0 Express board acts as the brains of the operation, since it has the perfect amount of pins to control the LCD, analog potentiometers and NeoPixels. The three potentiometers that are controlling the RGB values are connected to analog pins 0, 1 and 2. The NeoPixels' data pin is pin 13 and then LCD uses six of the digital pins (pins 5, 6, 9, 10, 11 and 12).



https://adafru.it/dJp

https://adafru.it/dJp

The main portion of the circuit is soldered to a perma proto board. This makes laying out the components very simple and allows for Ground and 5V to be easily accessible using the rails. The LCD screen is not soldered directly to the board. Instead a row of female headers are soldered in so that the LCD can be removed easily if needed and to lift it to the proper height for the top of the enclosure. The LCD needs an potentiometer to control the screen brightness, which is located to its left.

The individual 5mm LEDs are only receiving power and ground from the Feather M0 Express, so that they turn on whenever the board receives power. They're being used as labels to help identify which value the potentiometers are controlling.

RGB CircuitPython Code Using the NeoPixel Library

This project is coded in CircuitPython. It utilizes the Adafruit Character LCD and NeoPixel libraries, so make sure that they are loaded into your library folder on your board.

For a more complete walkthrough on how to use CircuitPython please reference this Learn guide. (https://adafru.it/cpy-welcome)

```
import math
import time

import adafruit_character_lcd
import board
import digitalio
import neopixel
from analogio import AnalogIn

lcd_rs = digitalio.DigitalInOut(board.D5)
lcd_en = digitalio.DigitalInOut(board.D6)
lcd_d7 = digitalio.DigitalInOut(board.D12)
lcd_d6 = digitalio.DigitalInOut(board.D11)
lcd_d5 = digitalio.DigitalInOut(board.D10)
lcd_d4 = digitalio.DigitalInOut(board.D9)
lcd_columns = 16
lcd_rows = 2

lcd = adafruit_character_lcd.Character_LCD(
    lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7, lcd_columns, lcd_rows
)

potR = AnalogIn(board.A0)  # pot pin for R val
potG = AnalogIn(board.A1)  # pot pin for G val
potB = AnalogIn(board.A2)  # pot pin for B val
pixpin = board.D13  # neopixel pin
numpix = 8

strip = neopixel.NeoPixel(pixpin, numpix, brightness=0.1)


def val(pin):
    # divides voltage (65535) to get a value between 0 and 255
    return pin.value / 257


while True:
    strip.fill((int(val(potR)), int(val(potG)), int(val(potB))))
    # int converts float val to an integer

    lcd.set_cursor(3, 0)
    # text at the top of the screen
    lcd.message('R + G + B =')
    lcd.set_cursor(2, 1)
    # str converts float val to a string, trunc removes decimal values
    lcd.message(str(math.trunc(val(potR))))
    lcd.set_cursor(6, 1)
    lcd.message(str(math.trunc(val(potG))))
    lcd.set_cursor(10, 1)
    lcd.message(str(math.trunc(val(potB))))
    time.sleep(0.5)
    # refreshes screen to send updated values from pots
    lcd.clear()
```

The goal of the code is to have the three potentiometers each control the red, green and blue values of the NeoPixels. The NeoPixels should update in real time according to the values read from the potentiometers and then those same values should appear on the LCD's screen.

The analog values produced by the potentiometers are converted to digital values by dividing the analog value by 257, giving a minimum value of 0 and a maximum value of 255. This value exists as a `float`. For the values to be able to be used as a value for the NeoPixels they need to be converted to integers with `int`.

For these same values to be used with the LCD screen, they need to be converted to strings, which is done with `str`. The only issue that remains is that the values will appear with their decimal points, making them incredibly long and unable to fit on the screen. The decimal is removed by using the truncate function (`math.trunc`) from the Python math library. Finally, `lcd_clear()` has to be called at the end of the loop so that the LCD refreshes to reflect the changing values from the potentiometers.

HSV CircuitPython Code Using the FancyLED Library

This piece of CircuitPython code utilizes the Adafruit Character LCD, FancyLED and NeoPixel libraries, so make sure that they are loaded into your library folder on your board.

https://adafru.it/AKx

https://adafru.it/AKx

You don't have to remain confined to RGB values though. The FancyLED library in CircuitPython allows you to use HSV, or hue, saturation and value, to affect the NeoPixel strip's color.

The code is very similar to the NeoPixel library RGB code that we just looked at. The goals are still the same: have the pots control the three values that affect the NeoPixel strip's color (in this case HSV) and have these values display on the LCD in real time.

```
import time

import adafruit_character_lcd
import adafruit_fancyled.adafruit_fancyled as fancy
import board
import digitalio
import neopixel
from analogio import AnalogIn

lcd_rs = digitalio.DigitalInOut(board.D5)
lcd_en = digitalio.DigitalInOut(board.D6)
lcd_d7 = digitalio.DigitalInOut(board.D12)
lcd_d6 = digitalio.DigitalInOut(board.D11)
lcd_d5 = digitalio.DigitalInOut(board.D10)
lcd_d4 = digitalio.DigitalInOut(board.D9)
lcd_columns = 16
lcd_rows = 2

lcd = adafruit_character_lcd.Character_LCD(
    lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7, lcd_columns, lcd_rows
)

potH = AnalogIn(board.A0)  # pot for hue
potS = AnalogIn(board.A1)  # pot for saturation
potV = AnalogIn(board.A2)  # pot for value
pixpin = board.D13  # NeoPixel pin
numpix = 8

strip = neopixel.NeoPixel(pixpin, numpix, brightness=1.0)


def val(pin):
    # divides voltage (65535) to get a value between 0 and 1
    return pin.value / 65535


def round_pot_h():
    # rounds decimal value to 2 decimal places
    return round(val(potH), 2)
```

```
def round_pot_s():
    return round(val(potS), 2)


def round_pot_v():
    return round(val(potV), 2)


while True:
    # calls for HSV values
    color = fancy.CHSV(val(potH), val(potS), val(potV))
    # converts float HSV values to integer RGB values
    packed = color.pack()
    # writes converted int values to NeoPixels
    strip.fill(packed)

    lcd.set_cursor(3, 0)
    # text at the top of the screen
    lcd.message('H + S + V =')
    lcd.set_cursor(1, 1)
    # sends the rounded value and converts it to a string
    lcd.message(str(round_pot_h()))
    lcd.set_cursor(6, 1)
    lcd.message(str(round_pot_s()))
    lcd.set_cursor(11, 1)
    lcd.message(str(round_pot_v()))
    time.sleep(0.5)
    # refreshes screen to display most recent pot values
    lcd.clear()
```

The first change is the division of the analog voltage value by itself (`pin.value / 65535`) so that the minimum value read is 0 and the maximum is 1, since that is the range of HSV values.

Next, are the functions `round_pot_h` , `round_pot_s` and `round_pot_v` . These functions return a rounded number for the HSV values. Since HSV uses a decimal range between 0 and 1, the numbers can be very long in length, which won't fit on the LCD screen. Here, the numbers are being rounded to 2 decimal places so that all three HSV values can fit comfortably on the screen. However, you can easily change the number of decimal places to suit your needs.

In the loop, the HSV values are assigned to `color` , which is then converted from float HSV values to integer RGB values using `pack()` from the FancyLED library. These integer values are then written to the NeoPixel strip using `strip.fill(packed)` from the NeoPixel library.

For the LCD portion, the only big change from the NeoPixel RGB value code is that functions are being used and called to round the HSV values. This was done to avoid memory allocation issues.

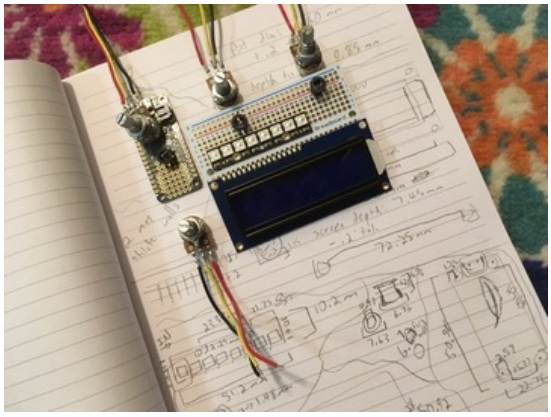
Also, you'll notice that the values are not being converted to integers for the LCD. This is because although we need integers to write the colors to the NeoPixel strip, we want to see the float HSV values on the screen. Just like the RGB value code, the rounded HSV values are sent to the LCD as strings with `str` .

Some examples of HSV values being dialed in.

3D Printing

Case Design

The case was designed in Fusion360 to have all of the components available on the top of the box with a hole in the back that allows the Feather M0 Express's USB port to be accessible. The bottom of the case has standoffs designed to screw the Feather and perma proto board in place with M2.5 screws. The case is also a snap fit case, making opening and closing it very simple.



Planning the case's layout and measurements

Printing

The case was printed with 20% infill and .2mm layer height. No supports are needed. Silver PLA filament was used to act as a neutral, yet still bright, color with the lights.

https://adafru.it/AJN

https://adafru.it/AJN

https://adafru.it/ALz

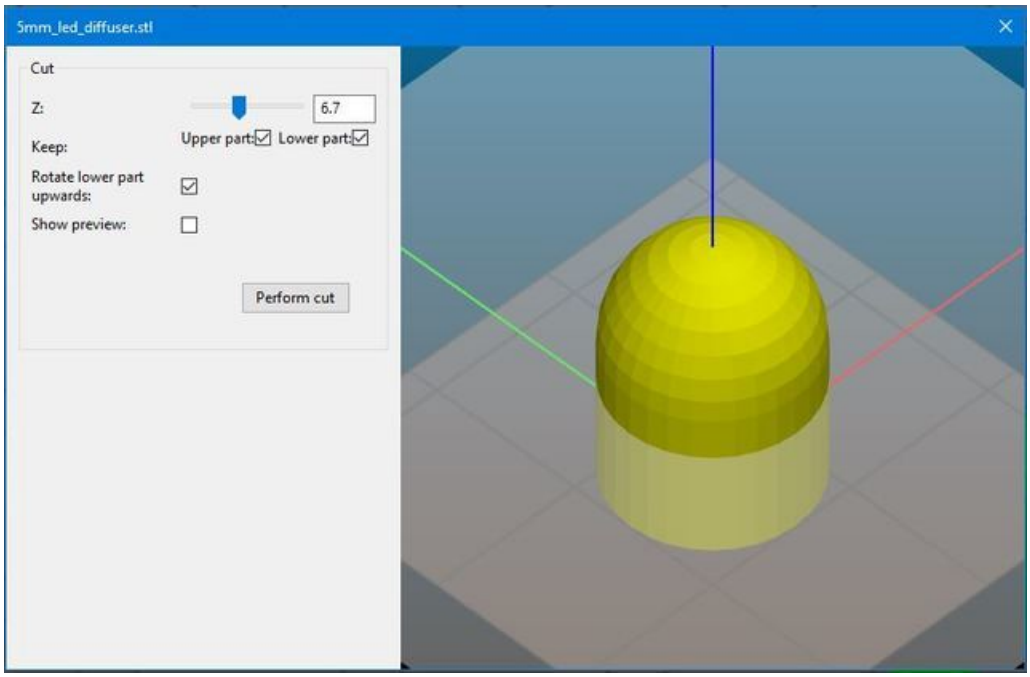
https://adafru.it/ALz

LED Diffusers

The single 5mm LEDs used are very bright and need to be diffused so that you can still look comfortably at the NeoPixel strip. There is a great diffuser model on Thingiverse designed by 3dprintsolutions that fits perfectly over a 5mm LED. But since the LEDs are in LED holders, they aren't fully exposed and as a result the provided model is too long. You can shorten the model though in your slicer by performing a cut. A cut at Z height 6.7 provided the best fit, sitting flush with the case. You may find that you need to print this a bit smaller, between 93%-95% since the lip found on the bottom of LEDs is not being used to catch onto the diffuser's housing.

https://adafru.it/AJO

https://adafru.it/AJO

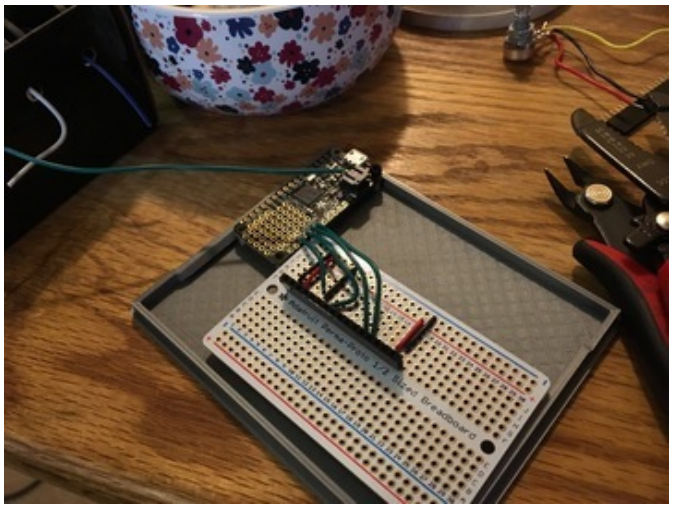


This cut model was then printed with 20% infill and .2mm layer height with white PLA. Supports were used as suggested by 3dprintsolution, the designer.

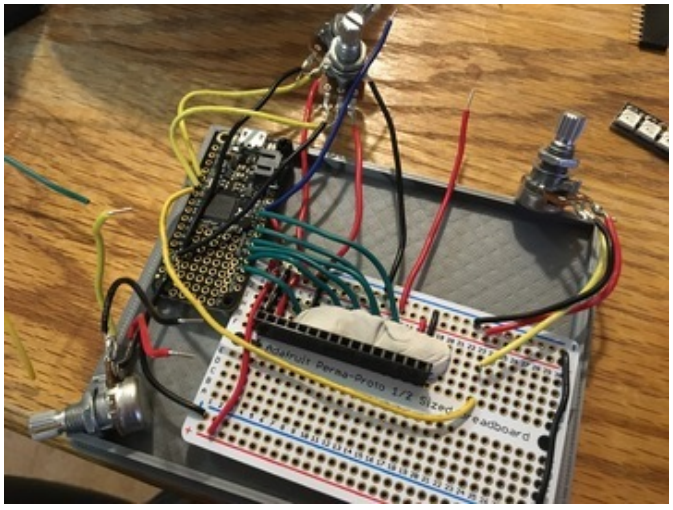
> It might be your first instinct to try clear PLA, but white PLA was found to diffuse the LEDs better for the purposes of this project since they are just for labeling purposes.

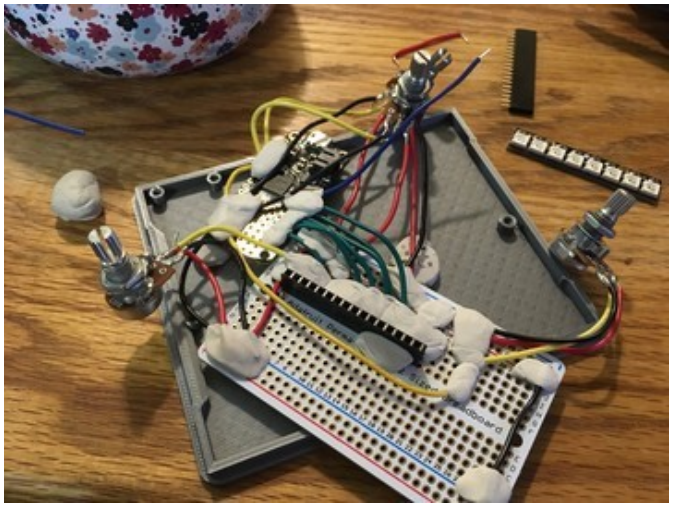
Assembly

Now it's time to solder everything together and fit everything snugly into the housing. Begin by placing all of the components onto the perma proto board and then running wires to the Feather M0 Express and other components. As mentioned in the Circuit portion of the guide, female headers were used for the LCD to plug into so that it can be removed easily and to lift it up high enough for the top of the case.



Circuit layout. Poster tack can be helpful when placing through-hole parts since it keeps everything in place and is easily removed after soldering.

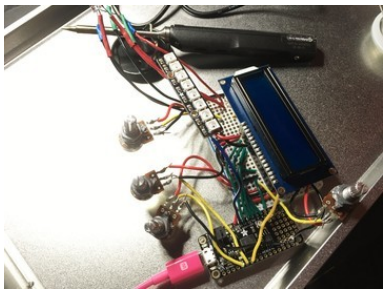




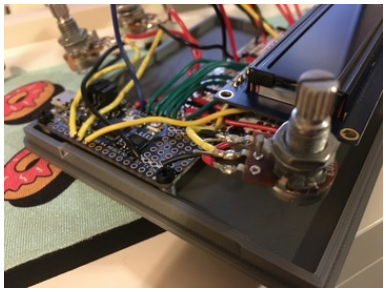
For the individual LEDs be sure to use heat shrink to avoid any shorts. You can use color coded heat shrink to help keep the clear LEDs straight.

When laying out wire and soldering, make sure you use long enough lengths of wire for the potentiometers, NeoPixel strip and individual LEDs so that when you insert them into the case they have enough room and none of their connections are being stressed.

After you've finished soldering and have tested everything, it's time to put everything into the housing. First, use M2.5 screws to secure the Feather M0 Express board and perma proto board to the bottom of the case. Also slot the 5mm LED holders into the top of the case. Next, feed the potentiometers and individual LEDs into their holes. Secure the potentiometers with their included washer. Then, press the NeoPixel strip into its slot. Depending on your printer's tolerances you may need to use a small piece of electrical tape to help keep it in place; otherwise the housing should hold the pixels securely. After that, you're ready to close up the case, which should snap shut. With the headers the LCD's screen should be flush with the top of the case and fit properly in its cutout without much adjustment.



Laying out parts after soldering



Finishing Touches

The knobs for the potentiometers help in labeling which color is being controlled. However, you may have noticed that there are no knobs with green markings. We can fix this with a simple Sharpie Paint marker. First, tape off the knob so that only the white strip is seen and then go to town with your Sharpie. You can use paint with a brush if you prefer, but the Sharpie is quick drying and less messy.

Taping and painting process for the green knob.

Once everything is inside the case you're ready to dial in all of the colors of the rainbow!


R + G + B =
88 38 81