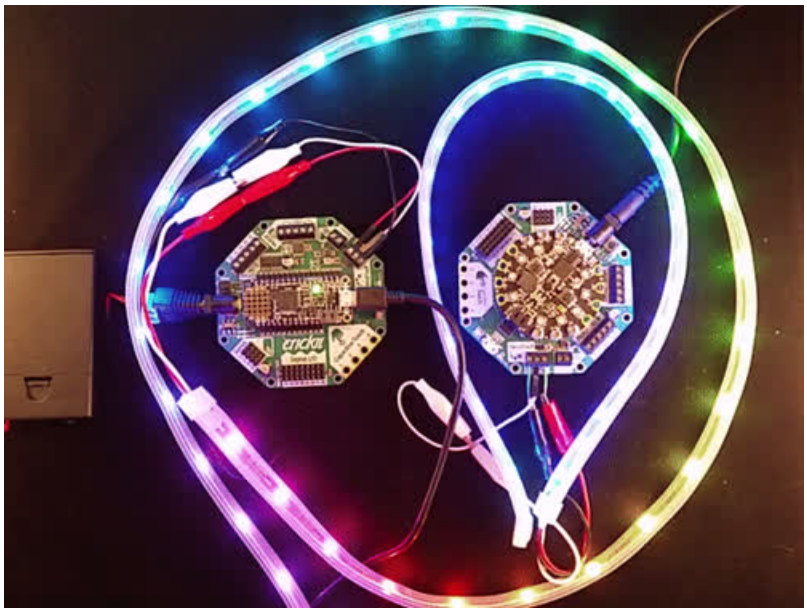
adafruit learning system

Make It Glow With Crickit

Created by Mike Barela



Last updated on 2018-09-17 10:15:11 PM UTC

Guide Contents

Guide Contents 2
Overview 3
- Different Types of Crickit Boards 3
- Parts List 3

Setting Up Your Programming Environment 5
- For MakeCode 5
- For CircuitPython 5

Using the Crickit NeoPixel Terminal Block 6
- Differences Between Crickit Boards 7

MakeCode 9
MakeCode 9
- Circuit Playground Express + Crickit with MakeCode 9
- Crickit FeatherWing 10
- micro:bit and Crickit 10

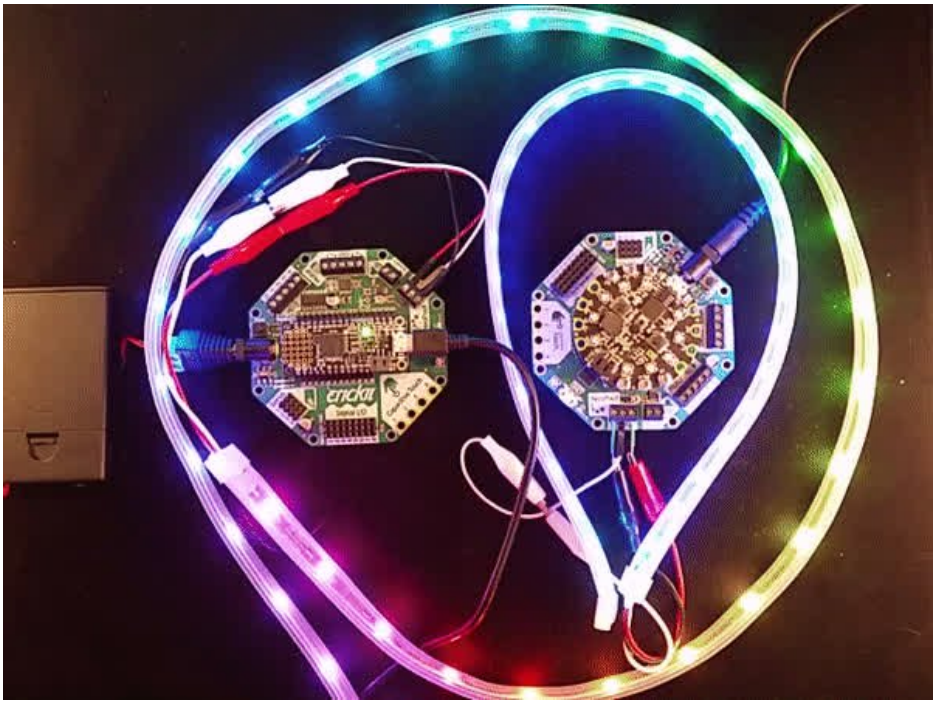
CircuitPython 12
- Crickit with Circuit Playground Express 12
- Crickit FeatherWing 14

Using Circuit Playground Express NeoPixels 16
- Using the on-Board Circuit Playground Express NeoPixels 16
- MakeCode 16
- CircuitPython 17

Using the Crickit Single LED 18
- MakeCode 18
- CircuitPython 19

You Want Even More NeoPixels? 20
- Memory 20
- Power and Current 20

Overview



Performing a search on robots of the past, most robots look unlit. Maybe some light for the eyes and not much else. Fortunately, anyone working with Adafruit Crickit (https://adafru.it/BTe), the Creative Robot and Interactive Construction KIT, can light their projects easily. There are numerous ways to light up your Crickit projects in only a couple of minutes.

This guide will show you how to use the NeoPixels on the Crickit and additional NeoPixels connected to the Crickit NeoPixel header. The code will be shown both in Microsoft MakeCode and in CircuitPython to allow you to integrate lights into your project quickly.

Whether you are looking for bling or serious environmental lighting, this is your starting point.

Different Types of Crickit Boards

There are several models of Crickit:

- Crickit (https://adafru.it/Biy) for Circuit Playground Express (https://adafru.it/wpF)
- Crickit (https://adafru.it/Cxy) FeatherWing for Feather Boards (https://adafru.it/BC6)
- Crickit (https://adafru.it/Cxz) for BBC micro:bit (https://adafru.it/yEO)
- Crickit HAT for Raspberry Pi (coming soon)

This guide will discuss NeoPixel connections in general for all boards and point out if anything is specific to a particular type of Crickit.

Parts List

1 x Crickit for Circuit Playground Express
Crickit - Creative Robotics and Interactive Construction Kit is an add-on to our popular Circuit

Playground Express that lets you #MakeRobotFriend using CircuitPython and MakeCode

OUT OF STOCK

---

1 x Circuit Playground Express

Circuit Playground Express is the next step towards a perfect introduction to electronics and programming.

ADD TO CART

---

1 x Crickit FeatherWing for any Feather

Crickit - Creative Robotics and Interactive Construction Kit is an add-on to our popular Circuit Playground Express that lets you #MakeRobotFriend using CircuitPython and MakeCode

ADD TO CART

---

1 x Feather M4 Express - Featuring ATSAMD51

Feather is powered by our new favorite chip, the ATSAMD51J19 - with its 120MHz Cortex M4

ADD TO CART

---

1 x Adafruit CRICKIT for micro:bit

Crickit - Creative Robotics and Interactive Construction Kit is an add-on to our popular Circuit Playground Express that lets you #MakeRobotFriend using CircuitPython and MakeCode

OUT OF STOCK

---

1 x BBC micro:bit Go Bundle

The newest and easiest way to learn programming and electronics - the BBC micro:bit.Designed specifically for kids and beginners, the micro:bit is a pocket-sized computer that you can code,

ADD TO CART

---

1 x NeoPixel Strip - 30 LEDs per Meter

This white strip can get you started with NeoPixels although other products work well also

ADD TO CART

---

1 x 5V 4A (4000mA) switching power supply

This switching supply gives a clean regulated 5V output at up to 4 Amps (4000mA). 110 or 240 volt input

ADD TO CART

Setting Up Your Programming Environment

If you are using Microsoft MakeCode or CircuitPython, you'll need to do a couple of basic things to set things up.

For MakeCode

In Makecode, the editor is located at https://makecode.adafruit.com/ (https://adafru.it/wmd)

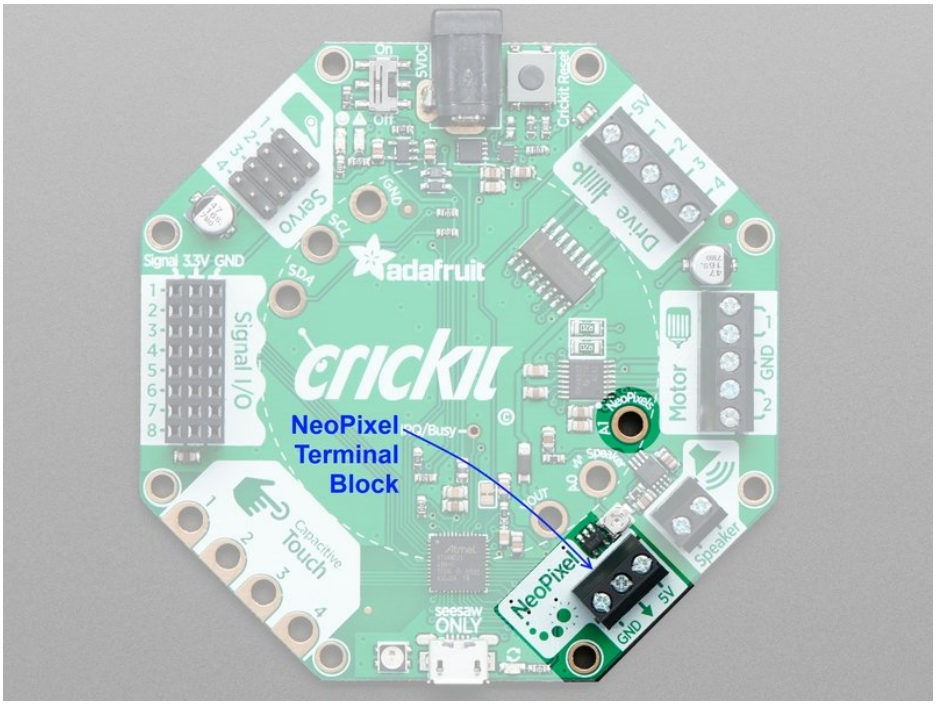
You will need to add the Crickit extensions. The Crickit Guide has the MakeCode information page (https://adafru.it/BKC) to install this extension.

For CircuitPython

See this guide page (https://adafru.it/cpy-welcome) from Introducing Adafruit Crickit on setting up your CircuitPython environment.

Adafruit suggests using the Mu editor to edit your code and have an interactive REPL in CircuitPython. You can learn about Mu and installation in this tutorial (https://adafru.it/ANO).

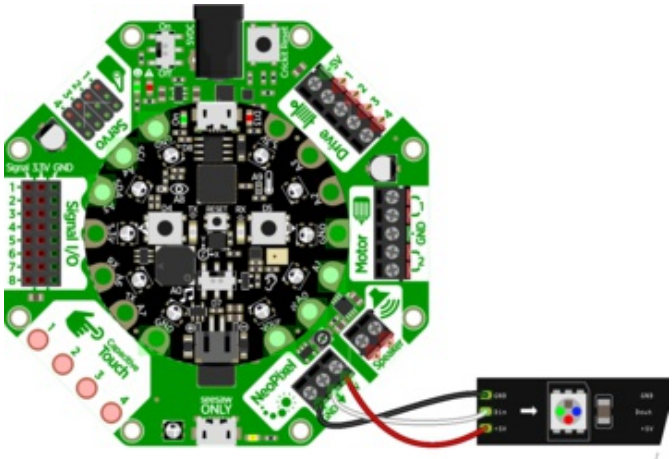
Using the Crickit NeoPixel Terminal Block



All Adafruit NeoPixel products have three main pins:

- a 5 volt pin, sometimes labeled **5**, **V**, **5V** or +
- a ground (GND) pin, sometimes labeled **GND** or **-**
- a Data In pin (someties labeled **DATA** or **DIN**).

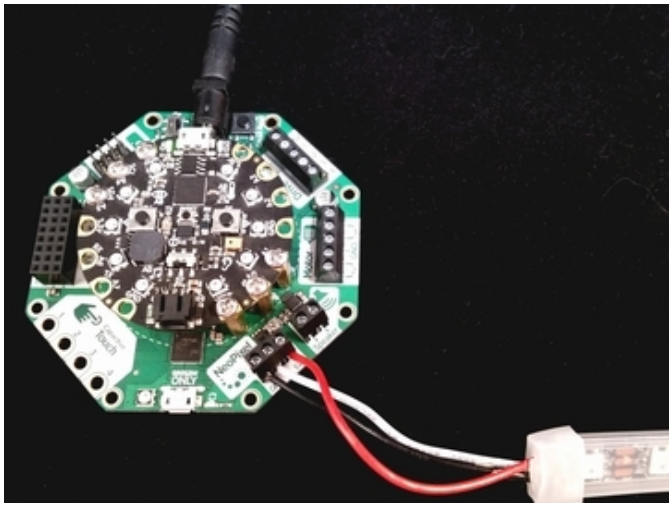
At the end of the NeoPixels or on the same board for a ring is a **Data Out (DOUT)** pin to connect to another NeoPixel product **Data In** pin if needed.

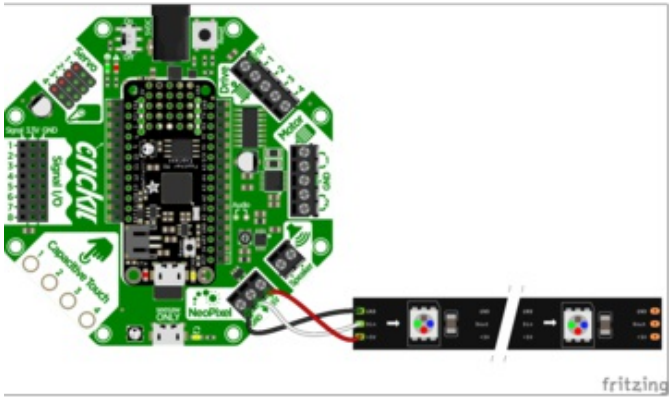


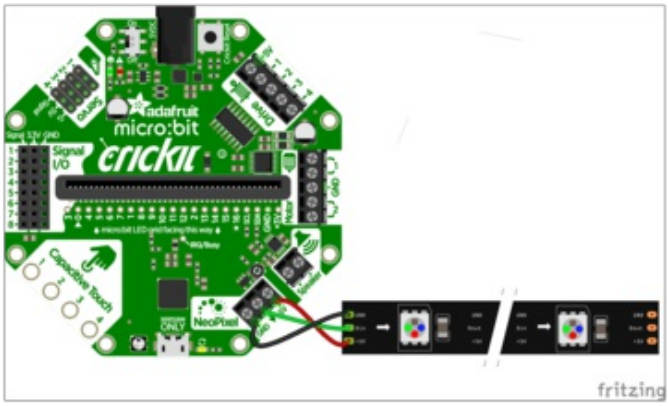
Crickit has a NeoPixel terminal block as shown at left. It works perfectly for NeoPixel strip connections.

Check carefully for the wire colors and wire ordering as they may vary from product to product. The markings on the products are always consistent so check those strips, rings, etc. for the pin names.

The examples in this guide will use a strip of NeoPixels. You are free to mix in strips, rings, etc. If you plan to use more than 100 NeoPixels in total, please read the page "You Want Even More (https://adafru.it/BTg) NeoPixels?" to get a feel for using a large number of pixels.







Note that you should not mix NeoPixels and Adafruit Dotstar products as they use separate types of wiring and software.

Differences Between Crickit Boards

This output is slightly different depending on what kind of Crickit you have:

- If you have a **Circuit Playground Crickit** then the NeoPixels are driven by the Circuit Playground **A1** pad by default. This way you can use the MakeCode emulator and built in Circuit Playground CircuitPython library. However, if you want, you can cut the jumper underneath the Crickit and solder closed the **ss** pad so that the seesaw chip controls the NeoPixels (for advanced hackers only).
- If you have a **Crickit FeatherWing** then the NeoPixels are driven by the seesaw chip on the Crickit. It's very similar to use of NeoPixels on other platforms. No extra pins are needed from your Feather. The NeoPixels are controlled by Crickit on seesaw pin #**20.**
- For the Crickit for micro:bit, **P16** is the pin for controlling the NeoPixels**.**

MakeCode

> MakeCode is for CircuitPlayground Express Crickit and micro:bit Crickit - it does not support Feathers at this time!

MakeCode

MakeCode has a large variety of code blocks which take advantage of the things you can do with NeoPixels. But in general, **THIS ONLY WORKS FOR THE CIRCUIT PLAYGROUND CRICKIT and Crickit for micro:bit**. The Feather Crickit NeoPixels are connected to the onboard seesaw chip so use in MakeCode isn't currently supported.

Circuit Playground Express + Crickit with MakeCode

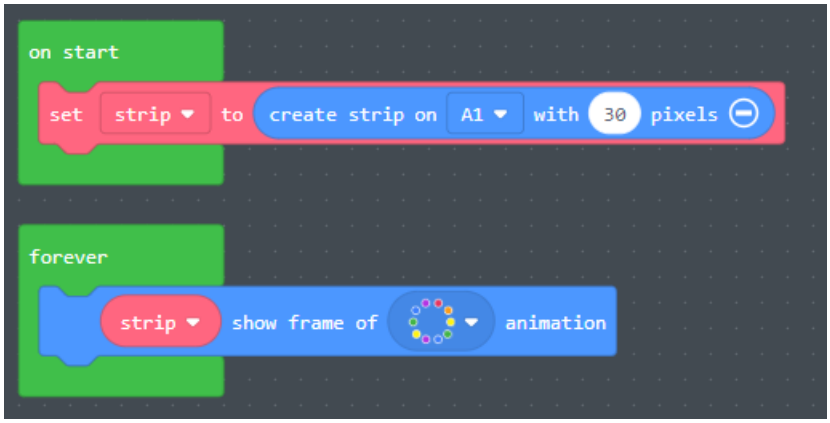
In the **LIGHT** block group, there is a special subgroup that pops below **LIGHT** when **LIGHT** is pushed called **... NEOPIXEL**. This provides a huge number of blocks to work with NeoPixels that are not on your Crickit or the microcontroller on Crickit like a Circuit Playground Express.

For Circuit Playground Express, the Crickit NeoPixel Terminal is connected to CPX Pin `A1`. When you use the MakeCode NeoPixel blocks to manipulate your externally connected NeoPixels, you need to use the **NEOPIXEL** subgroup block labeled `set strip to create strip`.

For the code below, I have connected a 30 NeoPixel strip (https://adafru.it/BPD) to the Crickit NeoPixel terminal block. When the program starts, the `on start` code up the variable named `strip` to refer to a NeoPixel strip connected to `A1` (which all Crickit strips are connected to) with `30` NeoPixels on it (You have to click the `+` on the block to specify the pin `A1` and add the number of NeoPixels.

Then the program shows the rainbow animation on the strip forever. You can change the animation type or do lots of

other things on your strip. It's that easy!



Crickit FeatherWing

There currently is no MakeCode support for the FeatherWing Crickit! Please use Arduino or CircuitPython for your Feather+Crickit needs

micro:bit and Crickit

For micro:bit, there is a small sun icon on Pin **P16** on Crickit to help you remember that is the pin connected for NeoPixels.

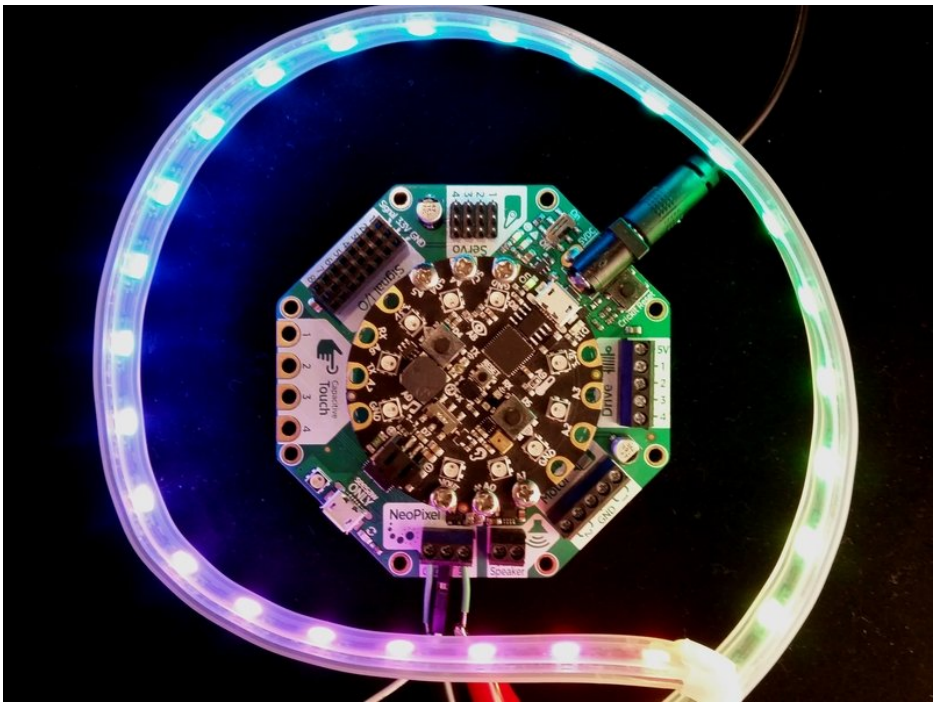
You need to add the NeoPixels extension to MakeCode for micro:bit for NeoPixel control. Click the **Advanced** button then **Add Package**. Select the *Adafruit NeoPixels* extension. You will now have a new code block group called **Neopixel** which has the blocks you want to control the NeoPixel strip.

The code below does what the above code does for CPX - creates a strip of 30 NeoPixels connected to Pin 16 and then displays a rainbow animation forever.

https://adafru.it/Cw0

https://adafru.it/Cw0

CircuitPython



The sample code below is a tiny bit different depending if you have a Crickit for Circuit Playground Express or Crickit FeatherWing.

Both have some functions to make some colorful animations. Feel free to change the lighting to suit your project.

To pick your own color values, see http://www.color-hex.com/ (https://adafru.it/BTh) to find your favorite red, green, and blue to place in your code!

Crickit with Circuit Playground Express

The **NeoPixel** terminal block is controlled by the Circuit Playground Express pad A1. The pad A1 definition is obtained by `import board`. Then the NeoPixel routine is from `import neopixel`.

Various animations are provided by `def`ined functions `wheel`, `color_chase` and `rainbow_cycle`. Various solid colors are then defined, you are free to use whichever colors you wish.

You can define a new color variable as a Python tuple with three values for red, green, blue, for example `WHITE = (255, 255, 255)`.

```
# Drive NeoPixels on the NeoPixels Block on Crickit for
#  Circuit Playground Express
import time
import neopixel
import board

num_pixels = 30  # Number of pixels driven from Crickit NeoPixel terminal

# The following line sets up a NeoPixel strip on Crickit CPX pin A1
pixels = neopixel.NeoPixel(board.A1, num_pixels, brightness=0.3,
                           auto_write=False)
```

```
                            auto_write=False)

def wheel(pos):
    # Input a value 0 to 255 to get a color value.
    # The colours are a transition r - g - b - back to r.
    if pos < 0 or pos > 255:
        return (0, 0, 0)
    if pos < 85:
        return (255 - pos * 3, pos * 3, 0)
    if pos < 170:
        pos -= 85
        return (0, 255 - pos * 3, pos * 3)
    pos -= 170
    return (pos * 3, 0, 255 - pos * 3)

def color_chase(color, wait):
    for i in range(num_pixels):
        pixels[i] = color
        time.sleep(wait)
        pixels.show()
    time.sleep(0.5)

def rainbow_cycle(wait):
    for j in range(255):
        for i in range(num_pixels):
            rc_index = (i * 256 // num_pixels) + j
            pixels[i] = wheel(rc_index & 255)
        pixels.show()
        time.sleep(wait)

RED = (255, 0, 0)
YELLOW = (255, 150, 0)
GREEN = (0, 255, 0)
CYAN = (0, 255, 255)
BLUE = (0, 0, 255)
PURPLE = (180, 0, 255)

while True:
    print("fill")
    pixels.fill(RED)
    pixels.show()
    # Increase or decrease to change the speed of the solid color change.
    time.sleep(1)
    pixels.fill(GREEN)
    pixels.show()
    time.sleep(1)
    pixels.fill(BLUE)
    pixels.show()
    time.sleep(1)

    print("chase")
    color_chase(RED, 0.1)  # Increase the number to slow down the color chase
    color_chase(YELLOW, 0.1)
    color_chase(GREEN, 0.1)
    color_chase(CYAN, 0.1)
    color_chase(BLUE, 0.1)
    color_chase(PURPLE, 0.1)

    print("rainbow")
    rainbow_cycle(0)  # Increase the number to slow down the rainbow
```

Crickit FeatherWing

The NeoPixel block signal wire is connected to the Crickit Seesaw control chip pin #20. The following code sets up an external 30 NeoPixel strip.

```
# Drive NeoPixels on the NeoPixels Block on Crickit FeatherWing
import time
from adafruit_crickit import crickit
from adafruit_seesaw.neopixel import NeoPixel

num_pixels = 30  # Number of pixels driven from Crickit NeoPixel terminal

# The following line sets up a NeoPixel strip on Seesaw pin 20 for Feather
pixels = NeoPixel(crickit.seesaw, 20, num_pixels)

def wheel(pos):
    # Input a value 0 to 255 to get a color value.
    # The colours are a transition r - g - b - back to r.
    if pos < 0 or pos > 255:
        return (0, 0, 0)
    if pos < 85:
        return (255 - pos * 3, pos * 3, 0)
    if pos < 170:
        pos -= 85
        return (0, 255 - pos * 3, pos * 3)
    pos -= 170
    return (pos * 3, 0, 255 - pos * 3)

def color_chase(color, wait):
    for i in range(num_pixels):
        pixels[i] = color
        time.sleep(wait)
        pixels.show()
    time.sleep(0.5)

def rainbow_cycle(wait):
    for j in range(255):
        for i in range(num_pixels):
            rc_index = (i * 256 // num_pixels) + j
            pixels[i] = wheel(rc_index & 255)
        pixels.show()
        time.sleep(wait)

RED = (255, 0, 0)
YELLOW = (255, 150, 0)
GREEN = (0, 255, 0)
CYAN = (0, 255, 255)
BLUE = (0, 0, 255)
PURPLE = (180, 0, 255)

while True:
    print("fill")
    pixels.fill(RED)
    pixels.show()
    # Increase or decrease to change the speed of the solid color change.
    time.sleep(1)
    pixels.fill(GREEN)
```

```
pixels.fill(GREEN)
pixels.show()
time.sleep(1)
pixels.fill(BLUE)
pixels.show()
time.sleep(1)

print("chase")
color_chase(RED, 0.1)  # Increase the number to slow down the color chase
color_chase(YELLOW, 0.1)
color_chase(GREEN, 0.1)
color_chase(CYAN, 0.1)
color_chase(BLUE, 0.1)
color_chase(PURPLE, 0.1)

print("rainbow")
rainbow_cycle(0)  # Increase the number to slow down the rainbow
```

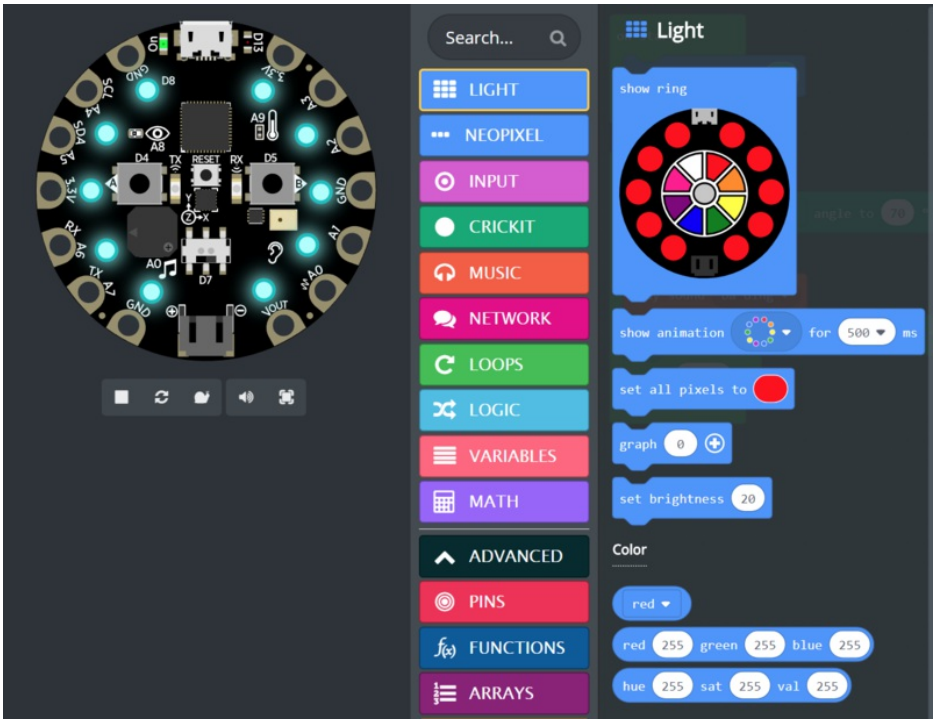
Using Circuit Playground Express NeoPixels

Using the on-Board Circuit Playground Express NeoPixels

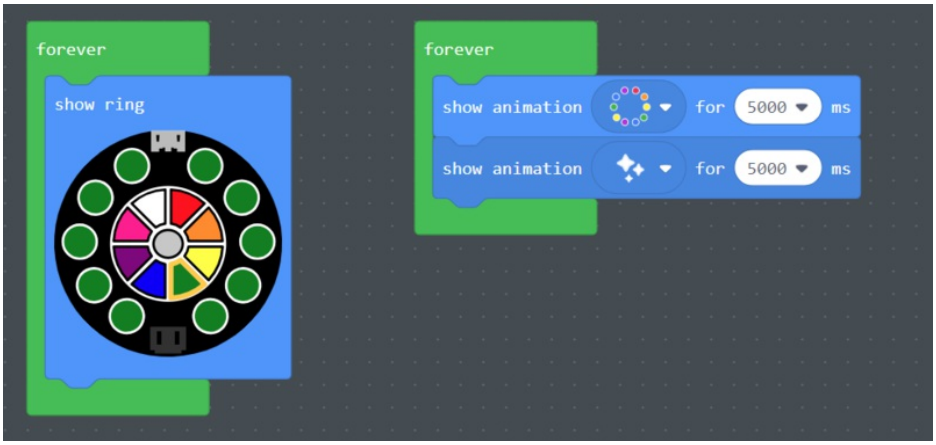
The Circuit Playground Express has ten Neopixels of its own that can be programmed independently from the Crickit single NeoPixel and Crickit NeoPixel terminal block connection.

MakeCode

For the Crickit NeoPixels we used the **NEOPIXEL** subgroup of the **LIGHT** block group in MakeCode. For the Circuit Playground Express, you use the blocks directly in the **LIGHT** block group, **not** the **NEOPIXELS** subblock blocks.



Below are two example programs (use one or another). The first just sets all the Circuit Playground Express LEDs to green. The second shows a rainbow animation for 5 seconds (= 5000 milliseconds), then shows a sparkle animation for another 5 seconds then loops back.



More on MakeCode and the Circuit Playground Express on the Introduction to Circuit Playground Express tutorial (https://adafru.it/AIX).

CircuitPython

It takes a bit more code to do NeoPixels in CircuitPython. The easiest way is to use the `adafruit_circuitplayground.express` library which sets everything up nicely. But the CPX library, as it's called, is large and when added to the Crickit support library `adafruit_crickit`, it does not leave a great amount of room for user programs.

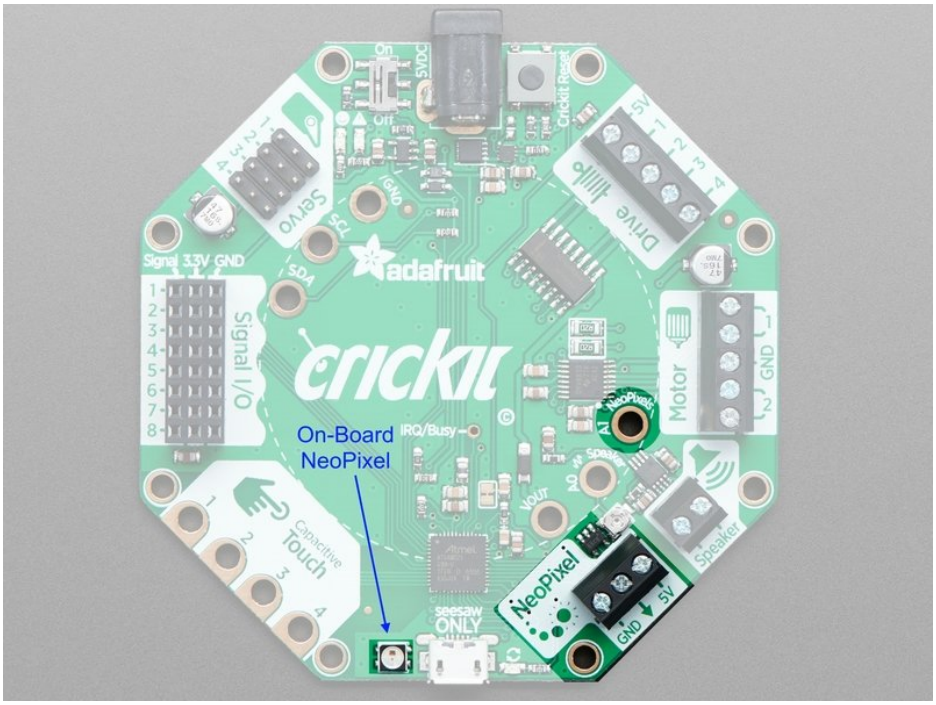
The same effects in the CPX library may be done with the `neopixel` library, it is just coded a bit differently.

The code below shows opening up the NeoPixels on-board and performing some red and blue color animation which you are free to change as you wish. The code for the strips to change color in previous examples will work here also with appropriate variable name changes for the NeoPixel object opened.

```
# Use the 10 NeoPixels on Circuit Playground Express via the
#   Adafruit neopixel library
import time
import neopixel
import board

# Set up the 10 Circuit Playground Express NeoPixels half bright
CPX_pixels = neopixel.NeoPixel(board.NEOPIXEL, 10, brightness=0.5)

# slowly power up via blue color
for i in range(50):
    CPX_pixels.fill((0, 0, i))
    time.sleep(0.05)

# blast off!
CPX_pixels.fill((255, 0, 0))

while True:
    # pulse effect
    for i in range(255, 0, -5):
        CPX_pixels.fill((i, 0, 0))
    for i in range(0, 255, 5):
        CPX_pixels.fill((i, 0, 0))
```

Using the Crickit Single LED

It isn't recommended that you use the Crickit NeoPixel LED because we use it to report when the battery is low! However, if you're curious, here's how to do it

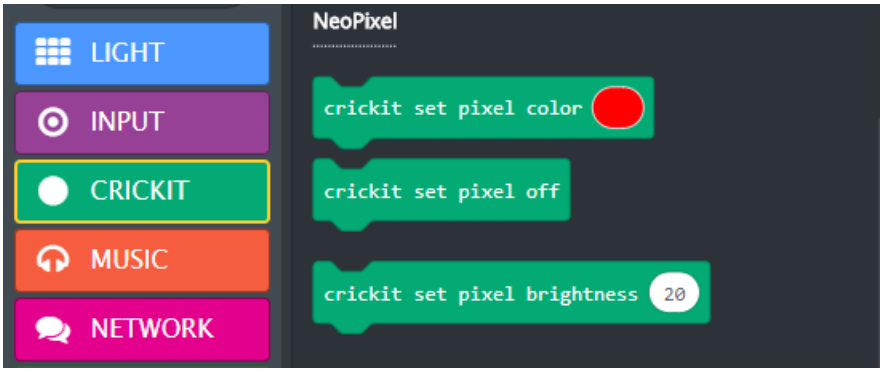


The On-board Crickit NeoPixel is great for providing status information to the user. It can also function as a light to help you find your bot in the dark.

This page will quickly show you how to use this NeoPixel.

MakeCode

You can use the three special NeoPixel blocks in the **CRICKIT** block group extension to change the single NeoPixel on-board Crickit. They work just like the NeoPixel blocks under the **LIGHT** block group but just for the one Crickit pixel.



You can independently set a color for the NeoPixel, turn it off, or set the brightness.

CircuitPython

The internal NeoPixel is on the Crickit's seesaw controller, pin #**27.** In addition to importing the `adafruit_crickit` library you'll need the `adafruit_seesaw.neopixel` library also.

```
# Using Crickit's onboard NeoPixel
# See http://www.color-hex.com/ for more colors and find your fav!
from adafruit_crickit import crickit
from adafruit_seesaw.neopixel import NeoPixel

crickit_status_pixel = NeoPixel(crickit.seesaw, 27, 1)  # one NeoPixel pin 27

# Fill them with our favorite color "#0099FF light blue" -> 0x0099FF
crickit_status_pixel.fill(0x0099FF)

while True:
    pass
```

The `fill` function will set the pixel to the hex code for red, green, and blue for the pixel. to turn the NeoPixel off, use a value of zero, such as `crickit_status_pixel.fill(0x000000)`.

You Want Even More NeoPixels?

Memory

The main limitation on use of NeoPixels on many microcontrollers is memory. The microcontroller, in general, has to reserve at least three bytes for the red, green, and blue pixel values. The more pixels to control, the more memory is used.

M0 based products like Circuit Playground Express and Feather M0 Express can usually drive about 100 pixels easily. With the Feather M4 Express, this is increased. The maximum will have to be something you test as with code, the amount of free RAM varies. 300 pixels may be possible on Feather M4.

If you plan to drive a large number of pixels, like for a whole room or art project, look at the full range Adafruit Products and guides. And catch us on the Adafruit forums under Glowy Things (https://adafru.it/BTi) and on the Adafruit Discord channel (). Folks can guide you to something if it is larger than can be accommodated with this introductory guide.

Power and Current

Crickit is designed to handle a decent amount of power (current) to your NeoPixel projects. Power = current x voltage, so with a constant 5 volts, power increases proportionally to current needs.

You will need to get an appropriate power supply to handle the peak (maximum) current you plan to draw in your project.

For a static (non-mobile) project, Adafruit recommends the 5V 2 Amp supply (https://adafru.it/BzI) for Crickit use.

If you plan on lots of motors and lots of NeoPixels, you can use a power supply up to 4 amps (https://adafru.it/e50). If you were to use 100 LEDs on full white, you might draw 4 amps according to Ladyada. If you do not run the LEDs all at full brightness and color, your project will use less current and save your eyeballs from the bright light.

If you don't run your NeoPixels full bright, say 30%, you may get up to 300 pixels running if you have enough memory free on the microcontroller you are using on your Crickit.

If you are running into limitations on the Crickit current use, you may have to look at a more sophisticated design. More often than not you'll run out of memory sooner than current capacity for the LEDs.