

# M.A.D.E.

## INTERMEDIATE E-TEXTILES GUIDES

FOR **MUSIC** | **ART** | **DESIGN** | **EXPERIENCES**

## CIRCUIT PLAYGROUND

## NEOPIXELS

Copyright © 2019 University of Pennsylvania



This work is licensed under the Creative Commons Attribution-NonCommercial 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/4.0/> or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

©2019 Google LLC All rights reserved. This curriculum module and research was supported by funding from Google's CS-ER program. Google and the Google logo are registered trademarks of Google LLC. The University of Pennsylvania is a recipient of Google's CS-ER program. The Google Grants program supports registered nonprofit organizations that share Google's philosophy of community service to help the world in areas such as science and technology, education, global public health, the environment, youth advocacy, and the arts. Google Grants is an in-kind advertising program that awards free online advertising to nonprofits via Google AdWords.

Any opinions, findings, and conclusions or recommendations expressed in this guide are those of the authors and do not necessarily reflect the views of Google, the University of Pennsylvania, Utah State University, University of Oregon, or Exploring Computer Science.

Please cite this work as Fields, D. A., Amely, J., Jayathirtha, G., Lindberg, L., Lui, D. & Kafai, Y. B. (2019). *M.A.D.E. Intermediate E-Textiles Guide: Circuit Playground NeoPixels*. Available online at <http://exploringcs.org/e-textiles/modules>.

## What you need to know first:

This module assumes that users have basic experience with making lighting patterns in Arduino (i.e., with `digitalWrite()` and `delay()` commands). For more introductory material, see the [Exploring Computer Science e-textiles curriculum unit](#), Buechley & Lui [Sew Electric](#), or other [introductory Arduino guides](#).

Other ECS E-Textiles modules as well as supporting code samples that may be referenced in this text can be found through <http://exploringcs.org/e-textiles/modules>.

## Table of Contents

### [Circuit Playground NeoPixels](#)

#### [1. NeoPixels - An Introduction](#)

##### [0.5 SUB-TASK #2 - Check for CircuitPlayground.h library](#)

##### [1.1 TASK #1 - NeoPixel Play](#)

##### [1.2 NeoPixel Challenges](#)

#### [2. NeoPixels - Managing Multiples](#)

##### [2.1 Explanation of the for\(\) Loop](#)

##### [2.2 TASK #2 - NeoPixels with a for\(\) Loop](#)

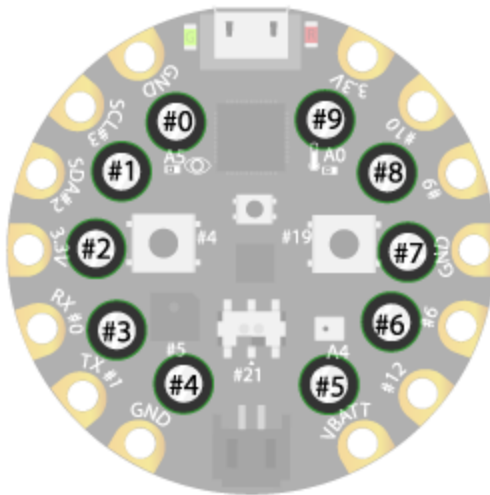
##### [2.3 CHALLENGES #3 - NeoPixels with a for\(\) Loop](#)

#### [3. More ways to use the Adafruit\\_CircuitPlayground.h Library](#)

See also: [Adafruit's Introduction to Circuit Playground | Library](#)

## 1. NeoPixels - An Introduction

NeoPixels are Red-Green-Blue individually controllable LEDs that can each be programmed for 16,777,216 different colors. They are the white boxes all around the Circuit Playground. Inside each box there are 3 sub-LEDs of red, blue, and green, as well as a little black chip that can control everything necessary to run all of the different colors and color patterns. Each NeoPixel is numbered—0 1 2 3 4 5 6 7 8 9—as its variable name from the top left, down and around to the top right. Code examples will have an *n* to let you know to put a *number* there.



Code to turn on a NeoPixel:

and this function

```
CircuitPlayground.setPixelColor(n, red, green, blue);
```

Use this library
Number of NeoPixel to light up
sub-LED colors (0 to 255)

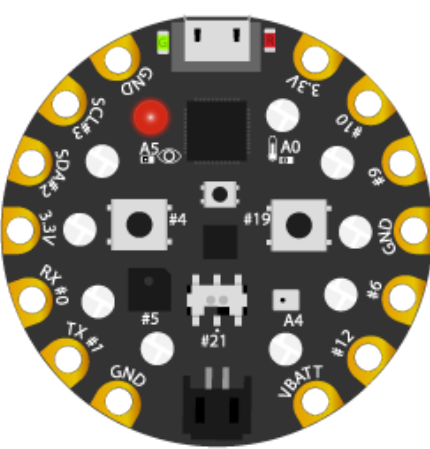
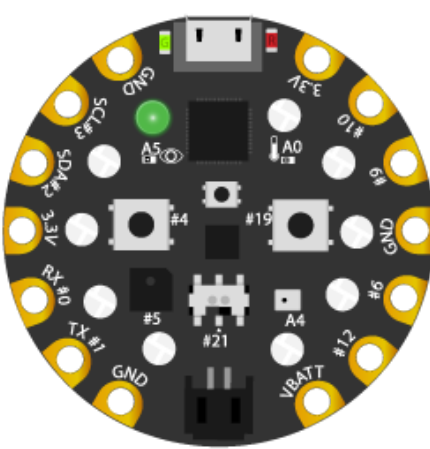
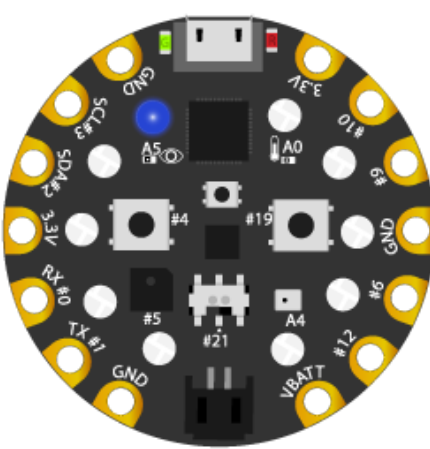
- The first variable in the parentheses is the number of the NeoPixel to turn on (0 through 9).
- The combination of the next 3 spaces are what make the color that you want. Much like mixing paint colors to get what you want, we are mixing levels of red, green, and blue sub-LED lights inside of the main NeoPixel LED to get a specific color. Each of these spaces can be set from 0 to 255.

# MADE - NEOPIXELS

EXAMPLE: To turn the first NeoPixel(0) bright red, you have to tell the Circuit Playground the number of the NeoPixel that you want to use, which is 0, and how much electricity to send to each sub-LED. The red sub-LED would be set all the way up to 255. The green is set get no electricity, at 0, and the blue is also set to 0, so these two sub-LEDs stay dark. Compare your code to the code below:

```
.....  
#include <Adafruit_CircuitPlayground.h> //includes the library  
  
void setup () {  
  CircuitPlayground.begin(); //tells the Circuit Playground to get going  
  CircuitPlayground.clearPixels(); //turns all the Neopixels off to start  
}  
  
void loop(){  
  CircuitPlayground.setPixelColor(0, 255, 0, 0); //turn it red!  
}  
  
.....
```

Result: One bright red Neopixel, with the blue and green parts off. See the table below for examples of the 3 main colors and the color code that goes with them.

<code>CircuitPlayground.setPixelColor(n,red,green,blue);</code>		
		
<code>CircuitPlayground.setPixelColor(0,255,0,0);</code>	<code>CircuitPlayground.setPixelColor(0,0,255,0);</code>	<code>CircuitPlayground.setPixelColor(0,0,0,255);</code>

### 0.5 SUB-TASK #2 - Check for CircuitPlayground.h library

1. Go to File > Examples > (Examples for Custom Libraries) Adafruit Circuit Playground > Hello\_CircuitPlayground > Hello\_Blink
2. Load code onto Circuit Playground.

If this step works for you, great! You already have the CircuitPlayground.h library installed. Continue to TASK #1. If this didn't work for you, try the [instructions Adafruit offers on installations](#).

### 1.1 TASK #1 - NeoPixel Play

1. Go to File > Examples > (Examples for Custom Libraries) Adafruit Circuit Playground > Hello\_CircuitPlayground > Hello\_NeoPixels
2. Load the code onto Circuit Playground.

Read through the code that you just loaded. Using the example code as a starting point, **Save As...** a new copy of the code to play around with. Tackle the NeoPixel Challenges below!

#### Remember!!!

Only change what is in the `void loop()`!

You **must** have the library, `#include <Adafruit_CircuitPlayground.h>` and the initialization code, `CircuitPlayground.begin()`; for the Circuit Playground to run your code!

### 1.2 NeoPixel Challenges

- EASY** Turn on one NeoPixel.
- EASY** Turn on every NeoPixel.
- EASY-ISH** Make the top right NeoPixel blink red.
- MEDIUM** Mixing the 3 color values, see if you can find these colors. Write down their RGB code. Experiment with different NeoPixel numbers. Remember to comment your code!

#### NeoPixel numbers and RGB Color Codes

```
Red = ( 0, 255, 0, 0); //this lights the 1st NeoPixel Red
Orange = ( , , , ); //
Yellow = ( , , , ); //
Lt Green = ( , , , ); //
Green = ( , , , ); //
Teal = ( , , , ); //
Blue = ( , , , ); //
Purple = ( , , , ); //
```

```
Pink = ( , , , ); //  
White = ( , , , ); //
```

- ❑ **MEDIUM** Start a New Sketch and light up only the NeoPixel next to the left button *and* the NeoPixel next to the right button. Make the colors Hot Pink and Teal. Make them blink just like the Hello\_NeoPixel example.
- ❑ **HARD** Start a New Sketch. Pull up the Hello\_Buttons code. It is all the way at the bottom of the Examples menu. (File > Examples > (Examples for Custom Libraries) Adafruit Circuit Playground > Hello\_CircuitPlayground > Hello\_Buttons). Use it to figure out how to make the left NeoPixel turn on when the Left button is pressed.
- ❑ **HARD** Add the ability to make the right NeoPixel turn on when the right button is pressed.
- ❑ **HARDER** Make it so that the right half of the Circuit Playground lights up Teal when the right button is pressed. Also make the left half of the Circuit Playground lights up Hot Pink when the left button is pressed.
- ❑ **HARDER** Make it so that when you load the code, the left half of the Circuit Playground lights up Teal and the right half of the Circuit Playground lights up Hot Pink *before* you touch any buttons. When you press the right button, the entire Circuit Playground should be Teal. When you press the left button, the entire Circuit Playground should light up Hot Pink.
- ❑ **HARDEST** Save As to make a duplicate of your button pressing code. Add the switch into your code so that if the switch is to the left, it behaves the same as before. But if the switch is to the right, change all of the colors and their interactions to be two other complimentary colors that interact the same way.

## 2. NeoPixels - Managing Multiples

So far you have played with turning on one or two NeoPixels at a time.

### Old Way

```
CircuitPlayground.setPixelColor(0, 255, 0, 0);
CircuitPlayground.setPixelColor(1, 255, 0, 0);
CircuitPlayground.setPixelColor(2, 255, 0, 0);
CircuitPlayground.setPixelColor(3, 255, 0, 0);
CircuitPlayground.setPixelColor(4, 255, 0, 0);
CircuitPlayground.setPixelColor(5, 255, 0, 0);
CircuitPlayground.setPixelColor(6, 255, 0, 0);
CircuitPlayground.setPixelColor(7, 255, 0, 0);
CircuitPlayground.setPixelColor(8, 255, 0, 0);
CircuitPlayground.setPixelColor(9, 255, 0, 0);
```

Results: 10 NeoPixels lit up red, 10 lines of code. That is a lot of repeated code to turn on all 10 NeoPixels, and repeated code means that there is a better way to do it.

A more efficient way to do this is with a `for()` loop to light up each NeoPixel in turn. The `for()` loop still completes so quickly that it looks like they all turn on at the same time!

### New Way with `for()` loop

```
void loop(){
  for (int i = 0; i < 10; i=i+1){
    CircuitPlayground.setPixelColor(i, 255, 0, 0);
  }
}
```

Results: 10 NeoPixels lit up red, 3 lines of code. (See below for more about `for()` loops.)

Luckily, it is much easier to **turn them all off** at the same time with:

```
CircuitPlayground.clearPixels();
```

Results: all NeoPixels off, 1 line of code.

### 2.1 Explanation of the `for()` Loop

There is a lot of stuff happening in a `for()` loop, and we will go through it line by line.

Line 1, `for()` loop:

```

    Initialization      Condition      Increment
      ↓                ↓              ↓
  for (int i = 0; i < 10; i=i+1){
  
```

Let's start with the three sections in the parentheses.

1. **Initialization**, `int i = 0`. This part says, "Create a variable and name it `i`. Set `i` to `0`, so that whenever you see `i`, think `0` (this will change!)."
2. **Condition** to be met, `i < 10`. This is how many times you want your code to repeat. As long as the equation (`i` is equal to **less than 10**) is **TRUE**, continue looping the code in the Action Block below. Once `i < 10` turns to **FALSE**, stop right here!
3. **Increment**, `i=i+1`. If you haven't stopped yet, add `1` to whatever number `i` is currently at. A shortcut for `i=i+1` is `i++` and it also increments by `1`.

Zooming back out of the parentheses: "**for**" the code to continue, the equation in the parenthesis has to be **TRUE**. So if it is still true, go to the **Action Block**—the code within the curly braces `{}`.

Line 2, the Action Block:

```

  CircuitPlayground.setPixelColor(i, 255, 0, 0);
}

```

The code here says, using the `Adafruit_CircuitPlayground.h` library, find and run the function `setPixelColor()`. This function lets you choose the Neopixel and the color you want it to be.

Important Parts:

- The `i` Variable  
This variable counts. We use it primarily to count how many times we have repeated the `for()` loop. We can use the same number that we get when we count repeats to light up the next Neopixel. So, we are using it for *two jobs*: the first job is to count how many times we have looped through the code, and the second job is to tell the code which Neopixel number to light up next.

The first time through is `0`, so we light up Neopixel `0`.

The second time through is `1`, so we light up Neopixel `1`.

The code will continue to light up Neopixels until we get to `9`—because `9` is less than `10` (`i < 10`)—but once we get to `10`, we stop repeating because `10` is not less than `10` and the equation is now **FALSE**.



- Sub-LED Brightness  
After `i`, the three numbers that follow are the brightness of the Red, Green, and Blue sub-LEDs. This code says to turn on only the red sub-LED and leave the green and blue ones off (`i, 255, 0, 0`). This is followed by the `for()` loop closing brace `}`.

Line 3:

```
}
```

This one is easy, BUT VERY IMPORTANT! The `void loop()` also has its set of curly braces, and if you forget to add the ending curly brace `}` for it, **your code won't work!** The same happens if you add too many ending curly braces.

What this looks like in use:

```
void loop(){  
  for (int i = 0; i < 10; i=i+1) {  
    CircuitPlayground.setPixelColor(i, 255, 0, 0);  
  }  
}
```

.....

In summary—here is sample code to turn all your Neopixels red. This has all the essential parts needed for Neopixel coding.

```
#include <Adafruit_CircuitPlayground.h> //includes the library  
  
void setup () {  
  CircuitPlayground.begin(); //tells the Circuit Playground to get going  
  CircuitPlayground.clearPixels(); //turns all the Neopixels off to start  
}  
  
void loop(){  
  for (int i = 0; i < 10; i=i+1) { //for loop set for 10 neopixels  
    CircuitPlayground.setPixelColor(i, 255, 0, 0); //turn them red!  
  }  
}
```

.....

## 2.2 TASK #2 - NeoPixels with a `for()` Loop

Using the code that you wrote to blink only one red NeoPixel as a starting point, build your own `for()` loop to turn on all the NeoPixels the color of your choice (or red is fine).

## 2.3 CHALLENGES #3 - NeoPixels with a `for()` Loop

For each Challenge, remember to start with a duplicate of your latest Challenge code by using **Save As**. You can also start from scratch—just remember to add `#include <Adafruit_CircuitPlayground.h>`

- ❑ **EASY** Add another `for()` loop to change all of the NeoPixels to a new color.
- ❑ **EASY** Keep adding `for()` loops until you have done all 10 main colors.
- ❑ **MEDIUM** Play with `i`. What happens if you add 2 instead of adding 1? What happens if you initialize (set `i` to...) to a number other than 0?
- ❑ **MEDIUM** Turn every *other* NeoPixel light on.
- ❑ **HARD** Make it so that you light up every NeoPixel Blue, and then every *other* NeoPixel Red. Remember to add a `delay()` or you won't see anything!
- ❑ **HARD** Can you figure out how to make a moving pattern? Hint: it is only in your mind that it looks like it is moving. In reality, the code lights up alternate NeoPixels, and then runs the same code but *with the 2 colors switched*. Use copy and paste to make alternating colors of Pink and White.
- ❑ **HARDEST** To really see the moving effect, you need three colors. Add green to your mix. Hint: you will have to think in sets of three for three colors, such as incrementing by 3, using three sections of similar code.
- ❑ **HARD** If you change where you use the `i` variable in your `for()` loop, instead of changing which NeoPixel lights up, you can change the amount of power that it sends to the sub-LED to create a fade on. Change your condition to go up to 255.
- ❑ **HARDEST** Create a fade of only Red, only Green, or only Blue. Can you fade on and off? Try combining them to get more colors in your rainbow. Can you capture all of the rainbow? Can you create theme fades, like the sunset, or the seasons?
- ❑ **HARDEST** Using the math function `random()`, declare `int red = random (256)` and play with it. What happens when you make random variables for Green And Blue and use all of them to set a NeoPixel color? Hint: Glitter.

### 3. More ways to use the Adafruit\_CircuitPlayground.h Library

This is just a tiny taste of all the functionality that the CP library holds. To learn more about the CircuitPlayground.h library and all of the options that it opens up, check out this [Circuit Playground Library Reference page](#) shared with Adafruit by user Caternusen.

To find more exciting projects and code made with the Circuit Playground, check out this link: <https://learn.adafruit.com/category/circuit-playground>