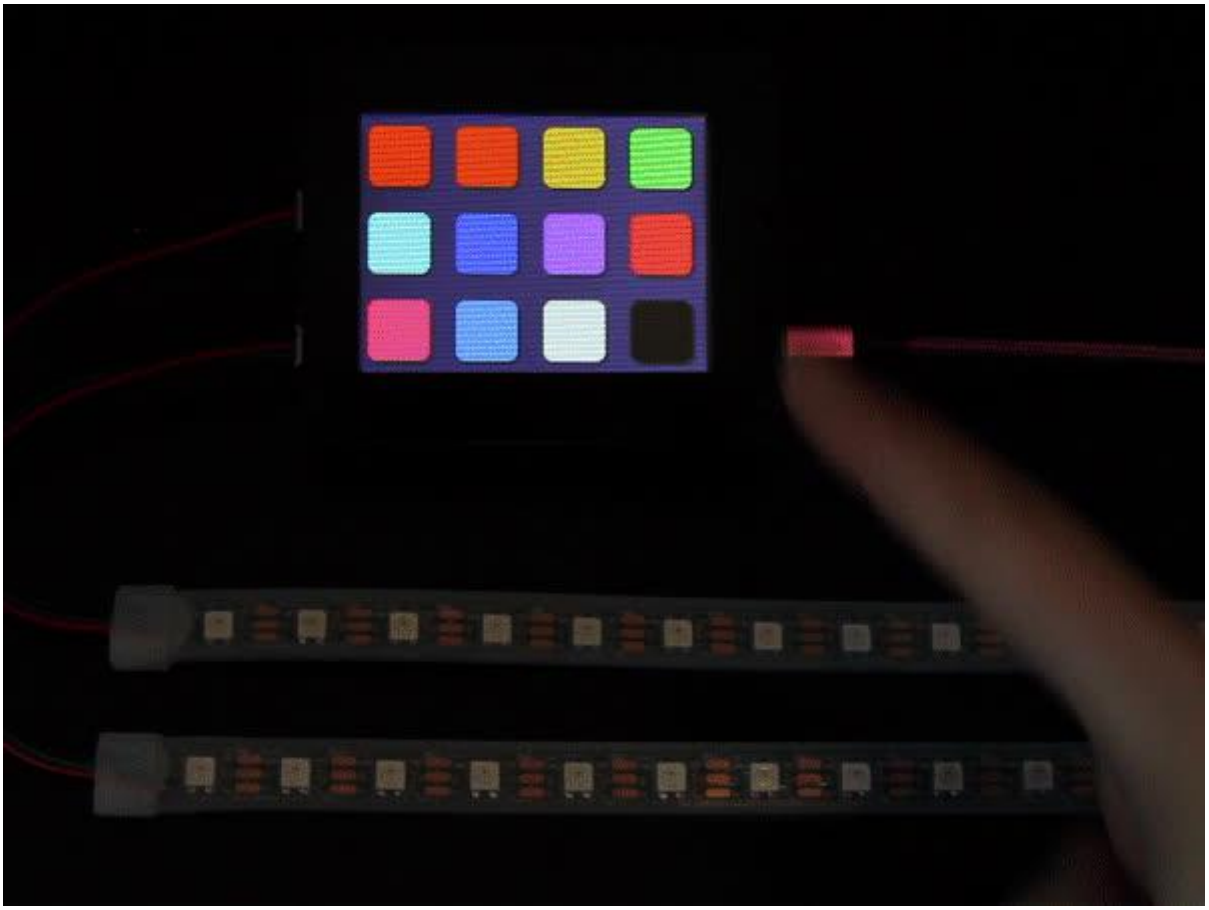
PyPortal NeoPixel Color Picker

Created by Kattni Rembor



https://learn.adafruit.com/pyportal-neopixel-color-oicker

Last updated on 2021-11-15 07:36:07 PM EST

Table of Contents

Overview 3

- Parts 4

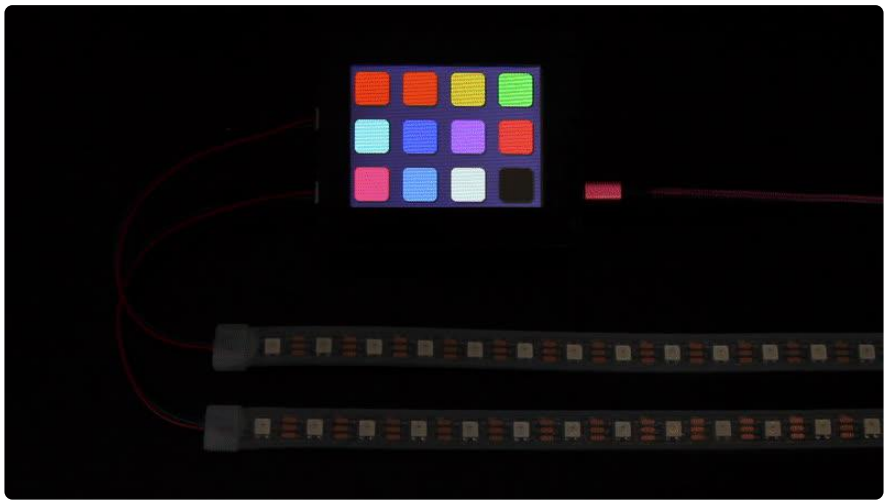
Install CircuitPython 5

- Set up CircuitPython Quick Start! 6
- PyPortal Default Files 8

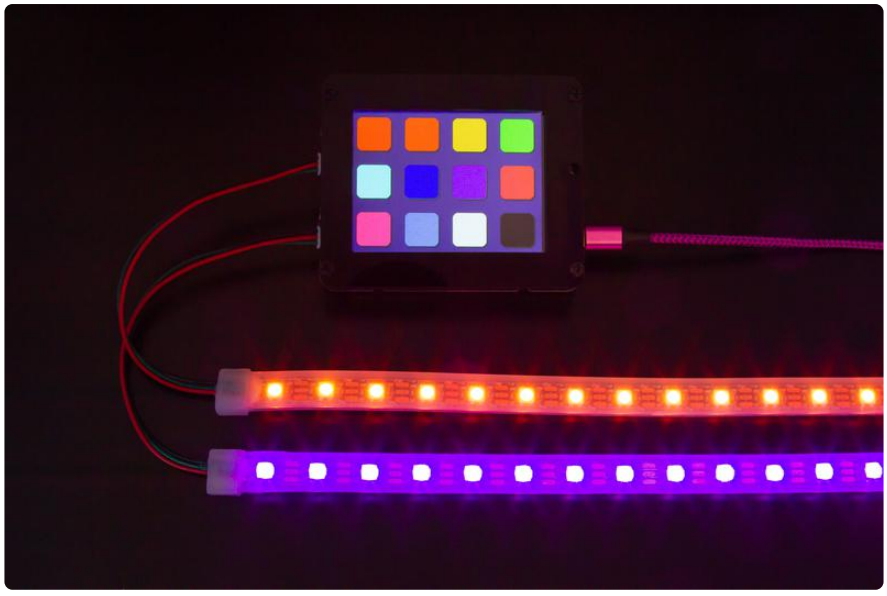
CircuitPython Code 8

- CircuitPython Libraries 9
- Secrets 10
- The Code 10

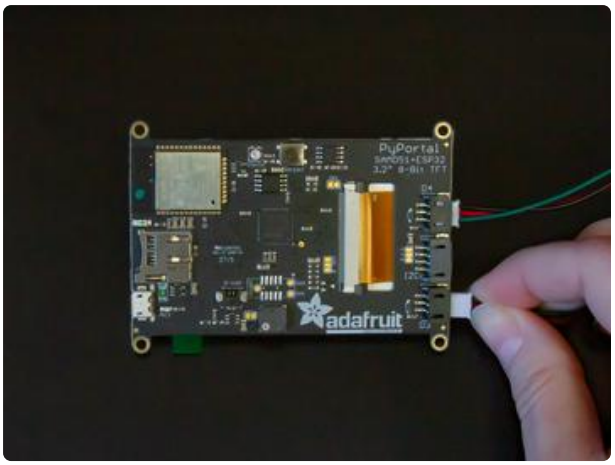
Overview

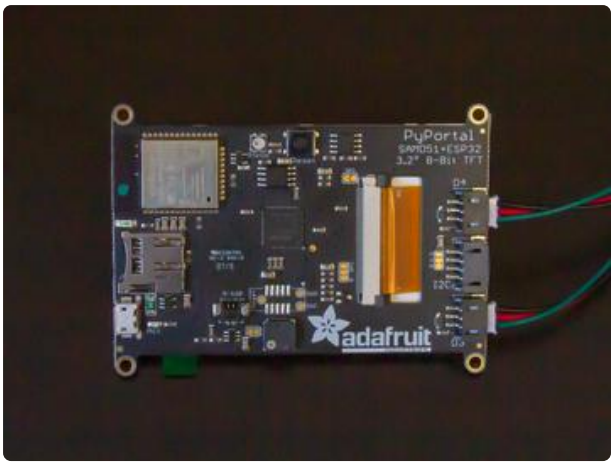


This simple project adds a little color to your life with CircuitPython, PyPortal and NeoPixels. Add two NeoPixel strips to your PyPortal, and display colored buttons that you can touch to set the LED colors. Wave your hand over the PyPortal light sensor to switch between controlling each strip separately or both together.



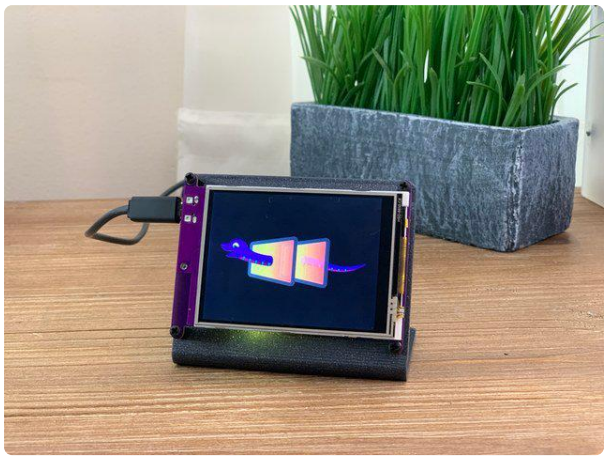
There's no soldering needed for this project! The PyPortal has two 3-pin JST connectors on it, labeled D3 and D4. These connectors work perfectly with 3-pin JST NeoPixel strips.





To build this project, simply plug one of these NeoPixel strips into each of the two connectors on the PyPortal.

Parts



Adafruit PyPortal - CircuitPython Powered Internet Display

PyPortal, our easy-to-use IoT device that allows you to create all the things for the “Internet of Things” in minutes. Make custom touch screen interface...
https://www.adafruit.com/product/4116

USB cable - USB A to Micro-B
This here is your standard A to micro-B USB cable, for USB 1.1 or 2.0. Perfect for connecting a PC to your Metro, Feather, Raspberry Pi or other dev-board or...
https://www.adafruit.com/product/592

5V 2A Switching Power Supply w/ USB-A Connector
Our 5V 2A USB power adapter is the perfect choice for powering single-board computers like Raspberry Pi, BeagleBone, or anything else that's power-hungry!This adapter was...
https://www.adafruit.com/product/1994

This project can control two of these strips:

Adafruit NeoPixel LED Strip with 3-pin JST PH Connector
Plug in and glow, this Adafruit NeoPixel LED Strip with JST PH Connector has 30 total LEDs in a "60 LED per meter" spacing,...
https://www.adafruit.com/product/3919

That's all there is to it! Now you're ready to set up the software on your PyPortal. Let's get started!

Install CircuitPython

CircuitPython (https://adafru.it/tB7) is a derivative of MicroPython (https://adafru.it/BeZ) designed to simplify experimentation and education on low-cost microcontrollers. It

makes it easier than ever to get prototyping by requiring no upfront desktop software downloads. Simply copy and edit files on the CIRCUITPY "flash" drive to iterate.

The following instructions will show you how to install CircuitPython. If you've already installed CircuitPython but are looking to update it or reinstall it, the same steps work for that as well!

Set up CircuitPython Quick Start!

Follow this quick step-by-step for super-fast Python power :)

Download the latest version of CircuitPython for the PyPortal via CircuitPython.org

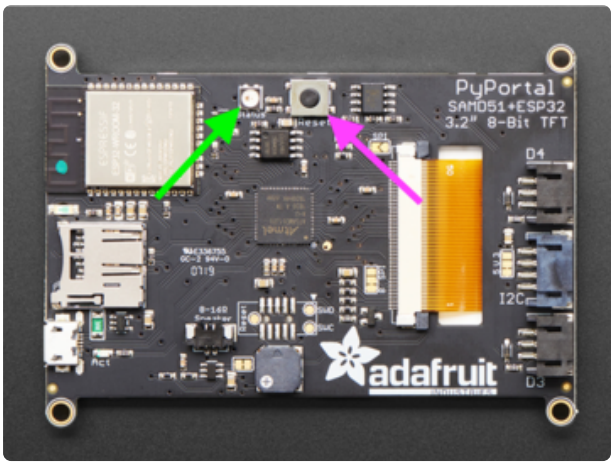
https://adafru.it/Egk

Download the latest version of CircuitPython for the PyPortal Pynt via CircuitPython.org

https://adafru.it/HFd

Click the link above to download the latest version of CircuitPython for the PyPortal.

Download and save it to your desktop (or wherever is handy).

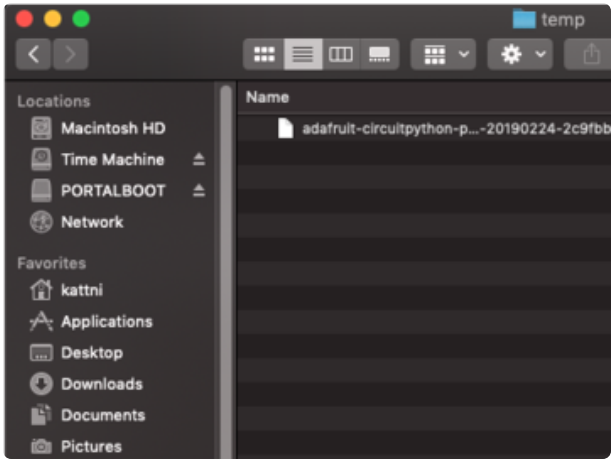


Plug your PyPortal into your computer using a known-good USB cable.

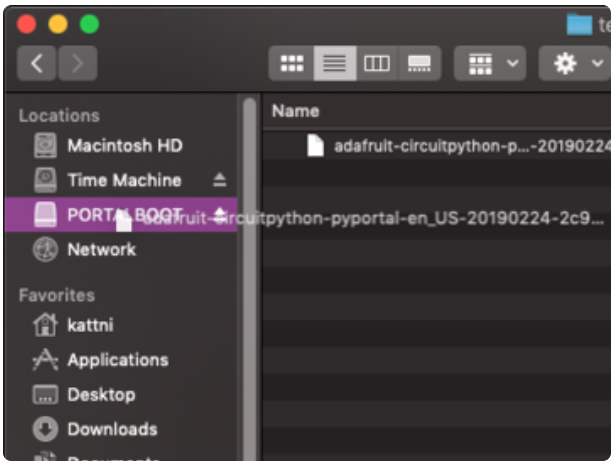
A lot of people end up using charge-only USB cables and it is very frustrating! So make sure you have a USB cable you know is good for data sync.

Double-click the Reset button on the top in the middle (magenta arrow) on your board, and you will see the NeoPixel RGB LED (green arrow) turn green. If it turns red, check the USB cable, try another USB port, etc. Note: The little red LED next to the USB connector will pulse red. That's ok!

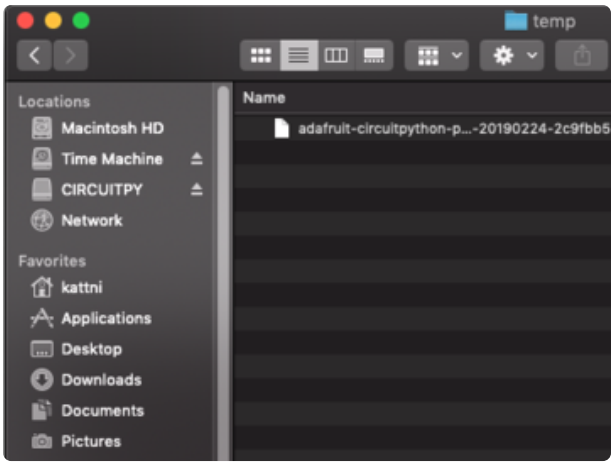
If double-clicking doesn't work the first time, try again. Sometimes it can take a few tries to get the rhythm right!



You will see a new disk drive appear called PORTALBOOT.



Drag the adafruit-circuitpython-pyportal-<whatever>.uf2 file to PORTALBOOT.



The LED will flash. Then, the PORTALBOOT drive will disappear and a new disk drive called CIRCUITPY will appear.

If you haven't added any code to your board, the only file that will be present is boot_out.txt. This is absolutely normal! It's time for you to add your code.py and get started!

That's it, you're done! :)

PyPortal Default Files

Click below to download a zip of the files that shipped on the PyPortal or PyPortal Pynt.

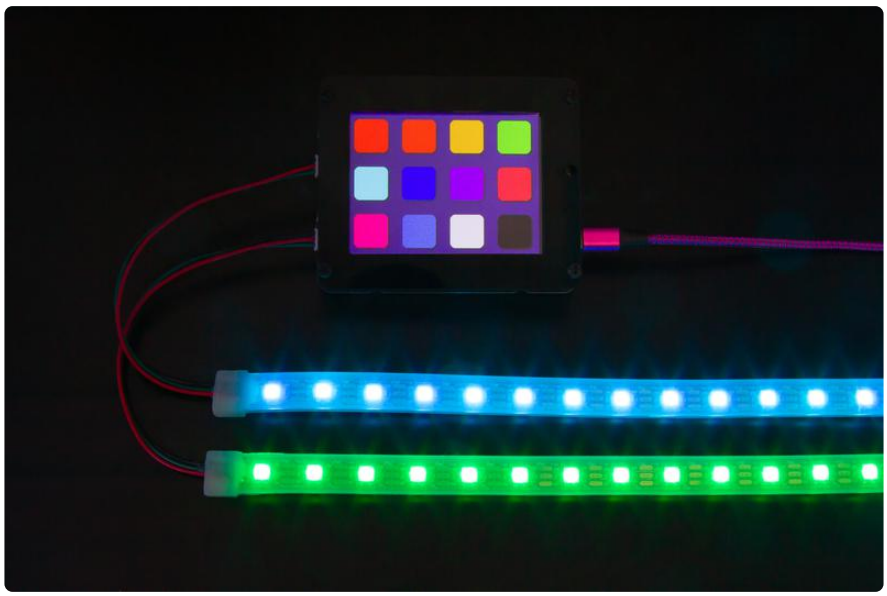
PyPortal Default Files

https://adafru.it/UF-

PyPortal Pynt Default Files

https://adafru.it/UGa

CircuitPython Code



This project is written in CircuitPython. The following will walk you through getting your PyPortal CIRCUITPY drive setup to run the code and give an overview of what's going on in the code.

This code requires some CircuitPython libraries to function. These libraries are not included with CIrcuitPython, so you'll need to load them yourself before the code will work.

CircuitPython Libraries

The first thing you'll need to do is load the necessary libraries onto your PyPortal. You'll copy the libraries individually from the bundle to your CIRCUITPY drive.

Latest CircuitPython Library Bundle Release

https://adafru.it/ENC

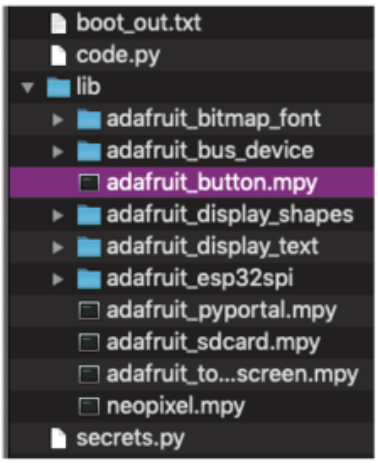
The libraries needed for this project are:

- Adafruit CircuitPython NeoPixel
- Adafruit CircuitPython PyPortal
- Adafruit CircuitPython SDCard
- Adafruit CircuitPython Touchscreen
- Adafruit CircuitPython Display Button
- Adafruit CircuitPython Bitmap Font
- Adafruit CircuitPython Bus Device
- Adafruit CircuitPython ESP32SPI
- Adafruit CircuitPython Display Shapes
- Adafruit CircuitPython Display Text

Download the latest CircuitPython Library Bundle (https://adafru.it/uap) to your computer and open the zip file. Find the `lib` folder. If copying the individual libraries, create a `lib` folder on your CIRCUITPY drive and copy the following files and folders into it:

- neopixel.py
- adafruit_pyportal.mpy
- adafruit_sdcard.mpy
- adafruit_touchscreen.mpy
- adafruit_button.mpy

- adafruit_bitmap_font
- adafruit_bus_device
- adafruit_esp32spi
- adafruit_display_shapes
- adafruit_display_text



Before continuing, ensure that you have at least the files neopixel.mpy, adafruit_pyportal.mpy, adafruit_sdcard.mpy, adafruit_touchscreen.mpy and adafruit_button.mpy, and the folders adafruit_bitmap_font, adafruit_bus_device, adafruit_esp32spi, adafruit_display_shapes and adafruit_display_text on your CIRCUITPY drive in the /lib folder. Once those are copied, you're all set to continue!

Secrets

The PyPortal has the ability to connect to your WiFi. Even though no network access is needed for this project, the PyPortal library still expects there to be a secrets.py file on your device. However, you do not need to include any credentials in the file for this project to work. The following is an example of a secrets.py file that will work with this project. Add the following file to your CIRCUITPY drive before continuing.

```
# This file is where you keep secret settings, passwords, and tokens!
# If you put them in the code you risk committing that info or sharing it

secrets = {
    }
```

The Code

This code sets a background color on your display, and then renders 12 colored buttons in a 4x3 grid. The color of the button is the color that you will set the NeoPIxels, e.g. the red button turns the NeoPixels on red. The light sensor is used as a toggle to switch between controlling the first strip, the second strip or both at the same time. Let's take a look!

```
import time
import board
```

```
from adafruit_pyportal import PyPortal
from adafruit_button import Button
import neopixel
import analogio

# Set the background color
BACKGROUND_COLOR = 0x443355

# Set the NeoPixel brightness
BRIGHTNESS = 0.3

light_sensor = analogio.AnalogIn(board.LIGHT)

strip_1 = neopixel.NeoPixel(board.D4, 30, brightness=BRIGHTNESS)
strip_2 = neopixel.NeoPixel(board.D3, 30, brightness=BRIGHTNESS)

# Turn off NeoPixels to start
strip_1.fill(0)
strip_2.fill(0)

# Setup PyPortal without networking
pyportal = PyPortal(default_bg=BACKGROUND_COLOR)

# Button colors
RED = (255, 0, 0)
ORANGE = (255, 34, 0)
YELLOW = (255, 170, 0)
GREEN = (0, 255, 0)
CYAN = (0, 255, 255)
BLUE = (0, 0, 255)
VIOLET = (153, 0, 255)
MAGENTA = (255, 0, 51)
PINK = (255, 51, 119)
AQUA = (85, 125, 255)
WHITE = (255, 255, 255)
OFF = (0, 0, 0)

spots = [
    {'label': "1", 'pos': (10, 10), 'size': (60, 60), 'color': RED},
    {'label': "2", 'pos': (90, 10), 'size': (60, 60), 'color': ORANGE},
    {'label': "3", 'pos': (170, 10), 'size': (60, 60), 'color': YELLOW},
    {'label': "4", 'pos': (250, 10), 'size': (60, 60), 'color': GREEN},
    {'label': "5", 'pos': (10, 90), 'size': (60, 60), 'color': CYAN},
    {'label': "6", 'pos': (90, 90), 'size': (60, 60), 'color': BLUE},
    {'label': "7", 'pos': (170, 90), 'size': (60, 60), 'color': VIOLET},
    {'label': "8", 'pos': (250, 90), 'size': (60, 60), 'color': MAGENTA},
    {'label': "9", 'pos': (10, 170), 'size': (60, 60), 'color': PINK},
    {'label': "10", 'pos': (90, 170), 'size': (60, 60), 'color': AQUA},
    {'label': "11", 'pos': (170, 170), 'size': (60, 60), 'color': WHITE},
    {'label': "12", 'pos': (250, 170), 'size': (60, 60), 'color': OFF}
    ]

buttons = []
for spot in spots:
    button = Button(x=spot['pos'][0], y=spot['pos'][1],
                    width=spot['size'][0], height=spot['size'][1],
                    style=Button.SHADOWROUNDRECT,
                    fill_color=spot['color'], outline_color=0x222222,
                    name=spot['label'])
    pyportal.splash.append(button.group)
    buttons.append(button)

mode = 0
mode_change = None

# Calibrate light sensor on start to deal with different lighting situations
# If the mode change isn't responding properly, reset your PyPortal to recalibrate
initial_light_value = light_sensor.value
while True:
```

```
    if light_sensor.value < (initial_light_value * 0.3) and mode_change is None:
        mode_change = "mode_change"
    if light_sensor.value > (initial_light_value * 0.5) and mode_change ==
"mode_change":
        mode += 1
        mode_change = None
        if mode > 2:
            mode = 0
        print(mode)
    touch = pyportal.touchscreen.touch_point
    if touch:
        for button in buttons:
            if button.contains(touch):
                print("Touched", button.name)
                if mode == 0:
                    strip_1.fill(button.fill_color)
                elif mode == 1:
                    strip_2.fill(button.fill_color)
                elif mode == 2:
                    strip_1.fill(button.fill_color)
                    strip_2.fill(button.fill_color)
                break
    time.sleep(0.05)
```

Setup

First you import the necessary libraries. Note that you're not importing all the libraries you copied to your CIRCUITPY drive - the libraries you do import rely on the ones that you don't.

Next there are a couple of variables you can set. Set `BACKGROUND_COLOR` to whatever you'd like for the background color using a hex color value. It defaults to a grey. Set `BRIGHTNESS` to the brightness you'd like for your NeoPixels, using a number `0` - `1` where the number represents a percentage brightness, e.g. `1` is 100%. It defaults to `0.3` or 30%.

```
# Set the background color
BACKGROUND_COLOR = 0x443355

# Set the NeoPixel brightness
BRIGHTNESS = 0.3
```

Then you setup the light sensor and the NeoPixel strips for use, and turn the NeoPixels off in the event that they were on.

Then you setup the PyPortal for use without networking. This is where the background color is set.

```
light_sensor = analogio.AnalogIn(board.LIGHT)

strip_1 = neopixel.NeoPixel(board.D4, 30, brightness=BRIGHTNESS)
strip_2 = neopixel.NeoPixel(board.D3, 30, brightness=BRIGHTNESS)
```

```
# Turn off NeoPixels to start
strip_1.fill(0)
strip_2.fill(0)

# Setup PyPortal without networking
pyportal = PyPortal(default_bg=BACKGROUND_COLOR)
```

Next is a series of variables that set the button colors using an RGB tuple, e.g. `(red, green, blue)`. In this case, `red`, `green` and `blue` are a whole number between `0` and `255`, where `0` is off and `255` is maximum. You can choose any colors you want. If you decide to change the variable names, you'll need to update them in the next section.

```
# Button colors
RED = (255, 0, 0)
ORANGE = (255, 34, 0)
YELLOW = (255, 170, 0)
GREEN = (0, 255, 0)
CYAN = (0, 255, 255)
BLUE = (0, 0, 255)
VIOLET = (153, 0, 255)
MAGENTA = (255, 0, 51)
PINK = (255, 51, 119)
AQUA = (85, 125, 255)
WHITE = (255, 255, 255)
OFF = (0, 0, 0)
```

The next section is a list that provides the information needed to create the buttons. Each line contains the following information for each button:

- The button label. In this case, they are numbered 1-12.
- The position of the button, using (x, y) coordinates.
- The size of the button, using (x, y) coordinates. Each button is 60 x 60 pixels.
- The color to set the button, using the color variables from above.

```
spots = [
    {'label': "1", 'pos': (10, 10), 'size': (60, 60), 'color': RED},
    {'label': "2", 'pos': (90, 10), 'size': (60, 60), 'color': ORANGE},
    {'label': "3", 'pos': (170, 10), 'size': (60, 60), 'color': YELLOW},
    {'label': "4", 'pos': (250, 10), 'size': (60, 60), 'color': GREEN},
    {'label': "5", 'pos': (10, 90), 'size': (60, 60), 'color': CYAN},
    {'label': "6", 'pos': (90, 90), 'size': (60, 60), 'color': BLUE},
    {'label': "7", 'pos': (170, 90), 'size': (60, 60), 'color': VIOLET},
    {'label': "8", 'pos': (250, 90), 'size': (60, 60), 'color': MAGENTA},
    {'label': "9", 'pos': (10, 170), 'size': (60, 60), 'color': PINK},
    {'label': "10", 'pos': (90, 170), 'size': (60, 60), 'color': AQUA},
    {'label': "11", 'pos': (170, 170), 'size': (60, 60), 'color': WHITE},
    {'label': "12", 'pos': (250, 170), 'size': (60, 60), 'color': OFF}
    ]
```

This information is then used to create the buttons by creating one button at a time and appending that button to the `buttons` list for later use.

```
buttons = []
for spot in spots:
    button = Button(x=spot['pos'][0], y=spot['pos'][1],
                    width=spot['size'][0], height=spot['size'][1],
                    style=Button.SHADOWROUNDRECT,
                    fill_color=spot['color'], outline_color=0x222222,
                    name=spot['label'])
    pyportal.splash.append(button.group)
    buttons.append(button)
```

Next, set the `mode` to `0` and set `mode_change = None`. This will be used in the toggle code.

Finally, you obtain the `initial_light_value` from the light sensor. This ambient light value is used to "calibrate" the light sensor to use as a toggle.

```
mode = 0
mode_change = None

initial_light_value = light_sensor.value
```

This "calibration" is necessary because initial testing found that a hard-coded value for the threshold of the toggle code meant that in different lighting situations, waving a hand over the sensor did not trigger the mode change properly. If it was too bright, then it never dropped below the hard-coded threshold even if the sensor was covered. If it was too dim, then it was constantly below the threshold and did not have the opportunity to trigger. Instead, the code "calibrates" on startup by obtaining an initial value, and then uses a percentage of that initial value as the thresholds. If you find that waving your hand over your PyPortal or covering the sensor is not triggering the mode change, you can recalibrate anytime by restarting your PyPortal.

Main Loop

The first part of the main loop is the toggle code. There are three modes: mode 0 which is controlling strip 1, mode 1 which is controlling strip 2, and mode 2 controlling both strips together. This code toggles through these modes in that order.

The amount of light reaching the sensor decreases as your hand begins to cover it until your hand is centered over it, and then it begins to increase again as your hand moves away from the sensor. The code uses this to create a toggle that only triggers once for each time your hand waves over the sensor.

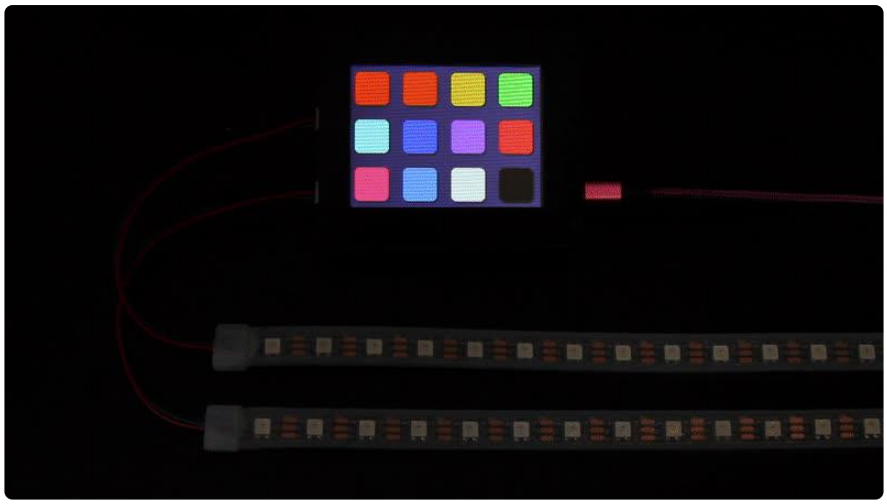
The current light value is compared to 30% of the initial light value, and when it drops below that threshold, the mode change sequence is started by setting `mode_change = "mode_change"`. As the current light value increases to greater than 50% of the

initial light value, the `mode` is increased by 1, and `mode_change` is set back to `None`. If the `mode` reaches greater than 2, it is set to `0`, to begin the cycle again. The `mode` is printed to the serial console as it changes.

```
if light_sensor.value &lt; (initial_light_value * 0.3) and mode_change is None:
    mode_change = "mode_change"
if light_sensor.value &gt; (initial_light_value * 0.5) and mode_change ==
"mode_change":
    mode += 1
    mode_change = None
    if mode &gt; 2:
        mode = 0
    print(mode)
```

The final section sets up touch on the display, and then checks to see if you've touched within the bounds of a button. If a particular button "contains" the point you've touched on the display, it prints `Touched` and the name of the button (its number, `1` - `12`) to the serial console. It checks to see which mode is currently active, and if it's `0`, it sets strip 1 to the button color, if it's `1`, it sets strip 2 to the button color, and if it's `3`, it sets both strips to the button color.

```
touch = pyportal.touchscreen.touch_point
if touch:
    for button in buttons:
        if button.contains(touch):
            print("Touched", button.name)
            if mode == 0:
                strip_1.fill(button.fill_color)
            elif mode == 1:
                strip_2.fill(button.fill_color)
            elif mode == 2:
                strip_1.fill(button.fill_color)
                strip_2.fill(button.fill_color)
            break
time.sleep(0.05)
```

That's what goes into creating a NeoPixel color picker with PyPortal and CircuitPython!

Try changing the colors and setting the strips to different combinations to fit the effect you're looking for. Have fun!