CircuitPython-Powered 3-minute Nightlight

Created by Dan Halbert

https://learn.adafruit.com/circuitpython-powered-gemma-nightlight

Last updated on 2023-08-29 03:46:16 PM EDT

Table of Contents

Overview

There are doorknobs with microcontrollers in them, so how about nightlights? We had houseguests, it was late, and I needed an extra nightlight in a hurry. I had all kinds of glowy things from Adafruit, but I didn't want to wire anything up. The RGB LED on a Gemma M0 was bright enough, and all I had to do was supply power. In one minute, I had a nightlight. Later I made it fancier.

Nightlight Basic

Note: When I first wrote this guide, using CircuitPython 3.x, the RGB LED on the Gemma M0 would light up with a solid color when a program was running. That is no longer the case with more recent versions of CircuitPython. However, you can download and install CircuitPython 3.1.2 for the Gemma M0, and the program below will work and make a nightlight.

If you want to use the Gemma M0 for anything else, such as the Nightlight Pro (), use the latest version of CircuitPython for the Gemma M0, which you can find here ().

CircuitPython 3.1.2 for the Adafruit Gemma M0

When you plug in a Gemma M0 running CircuitPython 3.1.2, the DotStar RGB LED on the board pulses green if there's no `code.py` program to run. If there is a `code.py`, it shows a solid color.

Pulsing green was a little spooky for this nightlight, so I wrote a `code.py` to run forever:

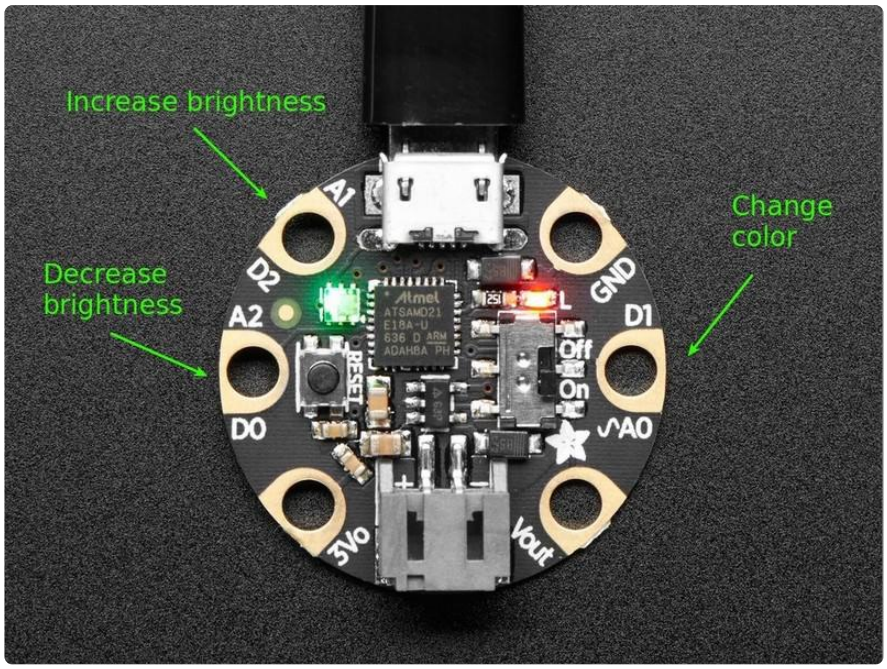
```
while True:
    pass
```

That's it, that's the whole nightlight program -- the shortest project code you'll ever see! To use, just copy and save it as `code.py` on your `CIRCUITPY` drive.

Then I plugged it in to an AC adapter with a short USB cable, but if you only have a long one, you could bunch it up with a twist tie.

This simple nightlight was good enough for our houseguests, but let's make a nicer one. Onward!

Nightlight Pro



Here's the Pro version of the nightlight. Instead of being limited to the plain green color at one intensity, you can change the colors and brightness of the RGB LED using the capacitive touch pads on the Gemma M0. Pads D0 and D2 on one side lower and raise the brightness, and D1 on the other side changes the color. You can tap the pads or hold them down. If you don't like the colors or the brightness range or steps in the program below, feel free to change them.

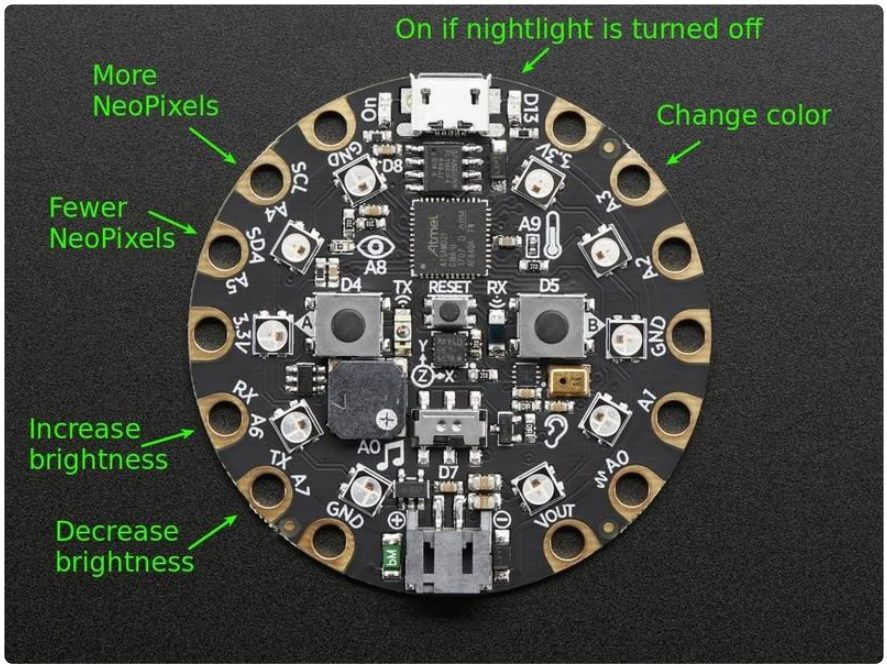
You need just one library, `adafruit_dotstar`. It's included in the Project Bundle you can download below.

Copy the program below, and save it as `code.py` on your `CIRCUITPY` drive.

```
# SPDX-FileCopyrightText: 2018 Dan Halbert for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time

import board
import touchio
import adafruit_dotstar

# Red, green, blue, and simple mixes of 2 or 3.
# Add your own choices here.
COLORS = (
    (0, 255, 0),
    (0, 0, 255),
    (255, 0, 0),
    (0, 255, 255),
    (255, 255, 0),
    (255, 0, 255),
    (255, 255, 255),
```

```
)

# The two left touch pads adjust the brightness.
# The right touch pad changes colors.
# Hold down or just tap.
brightness_down = touchio.TouchIn(board.D0)
brightness_up = touchio.TouchIn(board.D2)
change_color = touchio.TouchIn(board.D1)

dotstar = adafruit_dotstar.DotStar(board.APA102_SCK, board.APA102_MOSI, 1)
BRIGHTNESS_STEPS = 15
# Start at medium brightness, green.
brightness_step = 8
color_index = 0

while True:
    if brightness_down.value:
        # Don't go below 1.
        brightness_step = max(1, brightness_step - 1)

    if brightness_up.value:
        # Don't go above BRIGHTNESS_STEPS.
        brightness_step = min(BRIGHTNESS_STEPS, brightness_step + 1)

    if change_color.value:
        # Cycle through 0 to len(COLORS)-1 and then wrap around.
        color_index = (color_index + 1) % len(COLORS)

    # Scale brightness to be 0.0 - 1.0.
    dotstar.brightness = brightness_step / BRIGHTNESS_STEPS
    dotstar.fill(COLORS[color_index])

    time.sleep(0.2)
```

Nightlight Pro Deluxe CPX



If you have a Circuit Playground Express (CPX), here's an even fancier version of the nightlight program. It starts by lighting up two NeoPixels to be green. But you can change the number of NeoPixels, change their brightness, and change color, all via

the touch pads. It also provides a brightness limit (`MAX_BRIGHTNESS`), because the larger NeoPixels on the CPX are much brighter, and might be too bright for your nightlight.

Download the Project Bundle below, extract code.py from it, and upload code.py to CIRCUITPY. All the libraries you need are already included in CircuitPython for the CPX.

The program will use the CPX's built-in light sensor to turn off the nightlight if the ambient light is strong enough. If it turns off, it will turn on the small red LED to let you know it's still working. But I found the turn-off feature needs to be calibrated per room, so in the program below, that feature is turned off initially. Change the `TURN_OFF` value as described in the comments to enable it.

```
# SPDX-FileCopyrightText: 2018 Dan Halbert for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time

from adafruit_circuitplayground import cp

# Red, green, blue, and simple mixes of 2 or 3.
# Add your own choices here.
COLORS = (
    (0, 255, 0),
    (0, 0, 255),
    (255, 0, 0),
    (0, 255, 255),
    (255, 255, 0),
    (255, 0, 255),
    (255, 255, 255),
)

# Light level at which to turn off the nightlight.
# A typical level might be in the 20's or 30's. You'll need to experiment
# If you don't want the nightlight to turn off at all, set this to a large number
like 9999.
TURN_OFF = 9999

# The NeoPixels are really bright, so limit how bright they can get with
MAX_BRIGHTNESS.
# Increase its value if the nightlight is not bright enough, or decrease if it's
# too bright even at the lowest setting.
# MAX_BRIGHTNESS should be <= 1.0.
MAX_BRIGHTNESS = 0.5
BRIGHTNESS_STEPS = 15

# Start at a low brightness, green, 2 pixels,
brightness_step = 2
color_index = 0
num_pixels = 2
cp.pixels.auto_write = False

while True:
    if cp.light > TURN_OFF:
        # Indicate the nightlight is off.
        cp.red_led = True
        continue
    cp.red_led = False
```

```
    # Decrease brightness.
    if cp.touch_A7:
        # Don't go below 1.
        brightness_step = max(1, brightness_step - 1)

    # Increase brightness.
    if cp.touch_A6:
        # Don't go above BRIGHTNESS_STEPS.
        brightness_step = min(BRIGHTNESS_STEPS, brightness_step + 1)

    # Change color.
    if cp.touch_A3:
        # Cycle through 0 to len(COLORS)-1 and then wrap around.
        color_index = (color_index + 1) % len(COLORS)

    # Increase number of pixels.
    if cp.touch_A5:
        # Don't go below 1.
        num_pixels = max(1, num_pixels - 1)

    # Decrease number of pixels.
    if cp.touch_A4:
        # Don't go above 10 (number on Circuit Playgrounds).
        num_pixels = min(10, num_pixels + 1)


    # Scale brightness to be 0.0 - MAX_BRIGHTNESS.
    cp.pixels.brightness = (brightness_step / BRIGHTNESS_STEPS) * MAX_BRIGHTNESS
    for i in range(num_pixels):
        cp.pixels[i] = COLORS[color_index]
    for i in range(num_pixels, 10):
        cp.pixels[i] = 0
    cp.pixels.show()

    time.sleep(0.2)
```