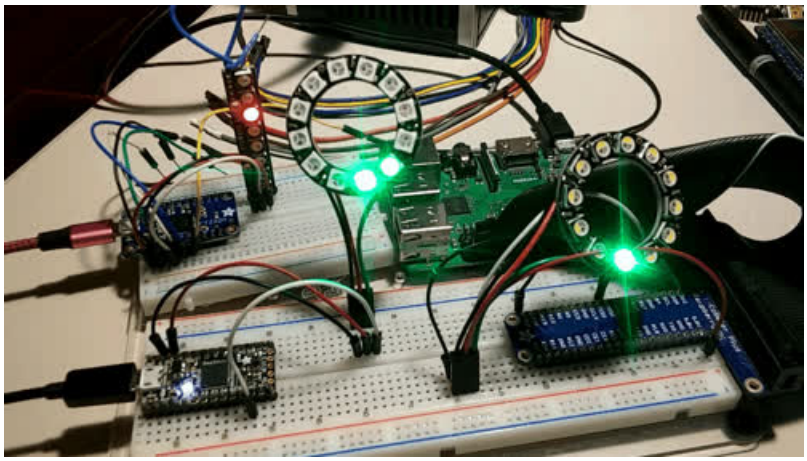
adafruit learning system

CircuitPython NeoPixel Library Using SPI

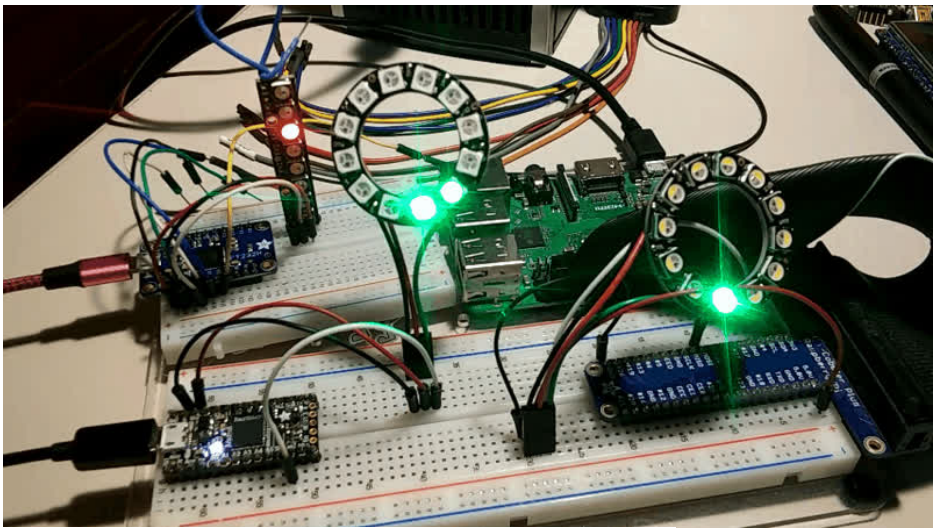
Created by Carter Nelson

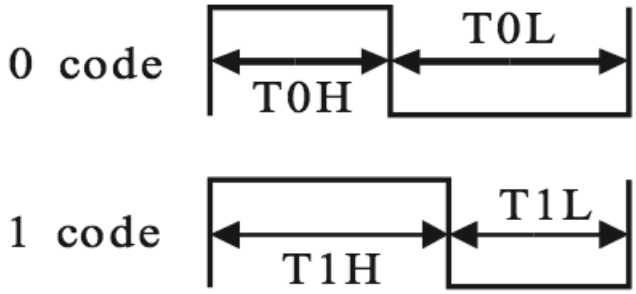


Last updated on 2021-01-28 12:25:12 PM EST

Guide Contents

Overview





NeoPixels are really great in terms of pin use. They only require three pins, and two of those are power and ground. They only need one pin for data which is used for **all** the NeoPixels attached. Amazing.

However, this comes at the cost of requiring the data signal maintain a very specific timing requirement. See here (https://adafru.it/jFu) for some details from the excellent NeoPixel Uberguide (https://adafru.it/oe1).

The NeoPixel SPI Hack

This is a pretty cool hack. The key enabling feature is the much faster relative speed of the SPI bus compared to the NeoPixel data signals. The NeoPixel data signal runs at 800kHz = 0.8MHz with some older ones running even slower at 400kHz = 0.4MHz. A SPI bus can be clocked in the 10s of MHz - **orders of magnitude faster than NeoPixel**!

This hack takes advantage of that faster speed to "synthesize" the NeoPixel data signal on the SPI's MOSI pin. In its most simple form, the hack turns every **bit** of NeoPixel data into a specific **byte** in the SPI data. There only two bytes that matter - one that represents a NeoPixel 0 bit, and one that represents a NeoPixel 1 bit.

Once the desired NeoPixel data has been translated into SPI data, it is simply clocked out on the SPI bus. The SPI bus frequency is set such that the data comes out at the expected NeoPixel timing. Then, by wiring the SPI MOSI pin to the NeoPixel data in pin, you can drive NeoPixels using the SPI bus. From the NeoPixel's point of view, it just sees the specific data signal it expects and you get happy blinky NeoPixels.

SPI Support in CircuitPython NeoPixel Library

All this work has been done for you via a new **NeoPixel_SPI** class that can be found in the CircuitPython NeoPixel SPI library (https://adafru.it/NHd). When driving NeoPixels via a SPI port, you simply use this class. After the initial setup, you can then use the NeoPixels as normal.

Let's see some examples.

FT232H Example

Using the SPI port option of an FT232H, we can drive NeoPixels from any PC with a USB port. See here for details about getting the FT232H installed and setup for your specific operating system:

https://adafru.it/Gbs

https://adafru.it/Gbs

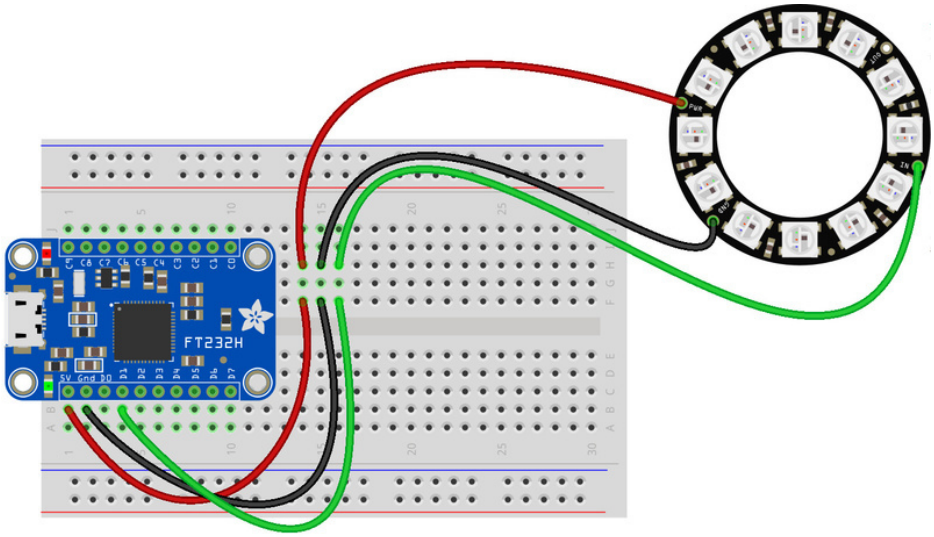
And of course, also install the NeoPixel library:

```
sudo pip3 install adafruit-circuitpython-neopixel-spi
```

FT232H Wiring

The wiring is pretty simple.

- **FT232H 5V** to **NeoPixel VIN**
- **FT232H GND** to **NeoPixel GND**
- **FT232H D1** to **NeoPixel DIN**

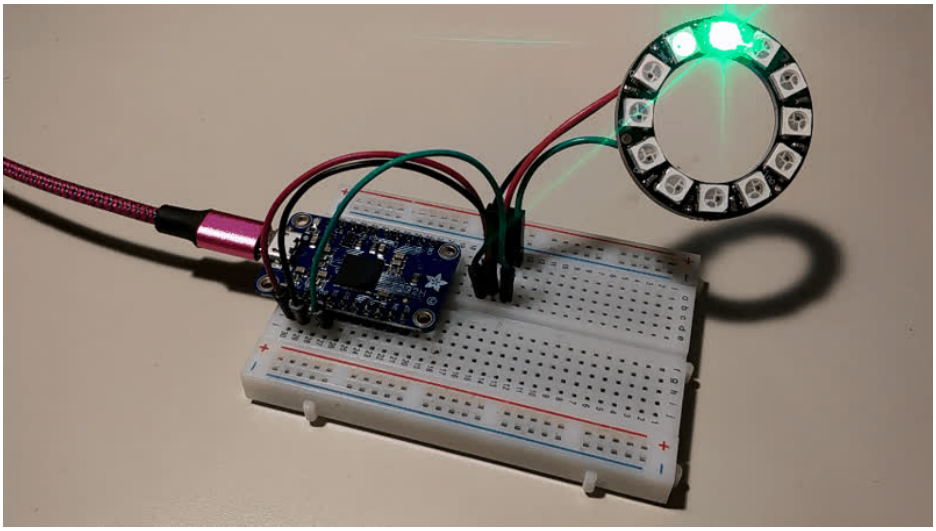


Note how only the **MOSI** (**D1**) pin of the SPI port is used. The other SPI pins, including SCLK, are not used at all.

Once you are wired up, try running the program in the **Example Code** section.

```
python3 neo_ring.py
```

Don't forget to set the BLINKA_FT232H environment variable. See guide linked above for OS specific details.



Example Code

Once you've gone through the setup specific to your hardware, you should be able to run this example code on any of them.

This example was written for the 12 pixel RGB NeoPixel ring (https://adafru.it/e8J) shown in the wiring diagrams. To use with other NeoPixel products, change **NUM_PIXELS** and **PIXEL_ORDER** to match your setup. You can also change or add colors to the **COLORS** tuple and change the speed via **DELAY**.

If you are using the FT232H, don't forget to set the BLINKA_FT232H environment variable.

Save this as **neo_ring.py**:

```
import time
import board
import neopixel_spi as neopixel

NUM_PIXELS = 12
PIXEL_ORDER = neopixel.GRB
COLORS = (0xFF0000, 0x00FF00, 0x0000FF)
DELAY = 0.1

spi = board.SPI()

pixels = neopixel.NeoPixel_SPI(spi,
                               NUM_PIXELS,
                               pixel_order=PIXEL_ORDER,
                               auto_write=False)

while True:
    for color in COLORS:
        for i in range(NUM_PIXELS):
            pixels[i] = color
            pixels.show()
            time.sleep(DELAY)
            pixels.fill(0)
```