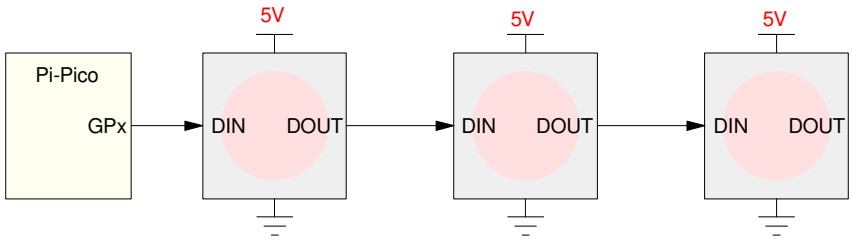
26. NeoPixels

Introduction:

NeoPixels are RGB LEDs with a single-wire interface. Several NeoPixels can be cascaded by connecting the DOUT pin to DIN of the next NeoPixel:

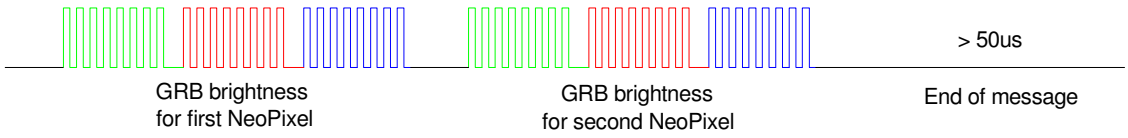


Wiring for NeoPixels: A single wire drives a string of LEDs

To drive a string of NeoPixels, a series of 24 bits is sent out in the order of (Green, Red, Blue):

- 255 is full brightness (20mA)
- 0 is off (0mA)
- Brightness (current) is proportional in-between

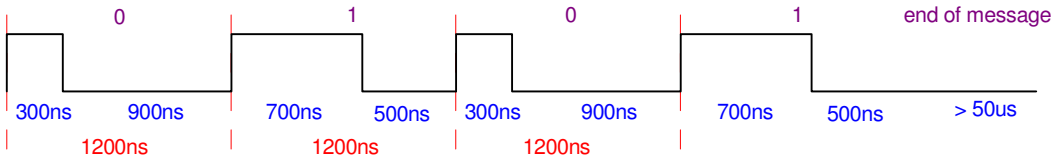
The first 24 bits drives the first neopixel, the next 24 bits drives the second, and so on. The end of the message is signified by an idle time of 50us or more.



To drive a string of NeoPixels, a series of 24 bits is sent out, one for each NeoPixel

Within each 24 bit message, logic 1 and 0 is defined by the pulse-width:

- Each bit is 1200ns
- Logic 0 has a 300ns high pulse
- Logic 1 has a 700ns high pulse



Encoding of logic 0 and 1 for NeoPixels

The function *bitstream()* in libraries *machine* and *neopixel* allows you to output 1's and 0's on an output pin with specific timing. The format for these commands are:

```
neopixel.bitstream(pin, encoding, timing, data, /)
machine.bitstream(pin, encoding, timing, data, /)
```

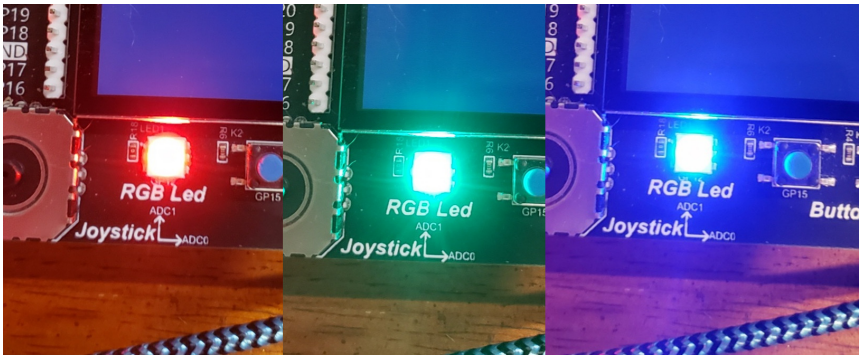
where

- pin: the GPIO pin to output the data
- encoding: 0 for high-low pulse duration. This transmits 0 and 1 bits as timed pulses starting with the most significant bit.
- timing: array of four times in nanoseconds: (high_0, low_0, high_1, low_1)
  - WS2812 at 800kHz would be (300, 900, 700, 500)
- data: binary array of data to send out

Example: Drive the NeoPixel on your board. Repeat the pattern (red, green, blue)

```
from machine import Pin, bitstream
from time import sleep

timing = [300, 900, 700, 500]
np = Pin(12, Pin.OUT)

red = bytearray([0,50,0])
green = bytearray([50,0,0])
blue = bytearray([0,0,50])

print(red)

while(1):
   bitstream(np, 0, timing, red)
   sleep(1)
   bitstream(np, 0, timing, green)
   sleep(1)
   bitstream(np, 0, timing, blue)
   sleep(1)
```

```
bytearray(b'\x00\x14\x00')
```

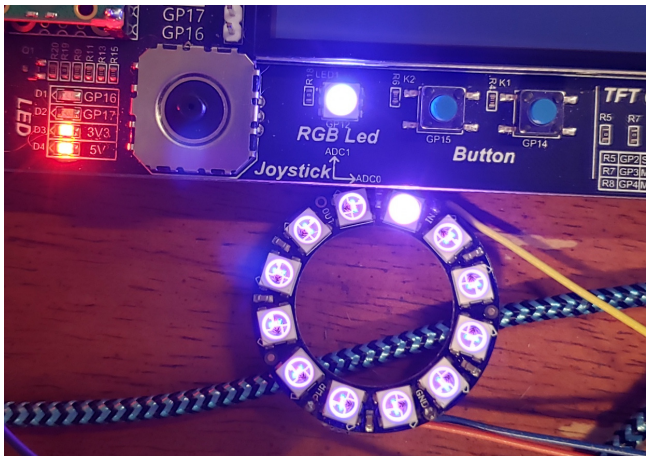


Output red / green / blue on the NeoPixel on the 52Pi board using bitstream

Example: Drive twelve NeoPixels attached to pin 12

```
from machine import Pin, bitstream
from time import sleep

timing = [300, 900, 700, 500]
np = Pin(12, Pin.OUT)

N = 12

X = bytearray([10,20,30])
for i in range(1,N):
    X.extend(bytearray([1,2,3]))

bitstream(np, 0, timing, X)
```

Program for driving twelve neopixels using *bitstream()*



Driving twelve neopixels using bitstream.
Pixel #0 is the bright one and matches the NeoPixel on the breadboard.

Note that X is an array of 3N bytes. If you want to change the color of the 4th NeoPixel, you can manually change that value. For example, making the 4th NeoPixel orange would be

```
X[12] = 100   # green
X[13] = 200   # red
X[14] = 0     # blue

bitstream(np, 0, timing, X)RGnp
```

Once set up, you can change the color of pixel N by updating the binary array
X[3N] = green, X[3N+1] = red, X[3N+2] = blue

Changing element {12, 13, 14} sets the color of the 4th NeoPixel

Using the Library *neopixel*

There is also a neopixel library which has the function

```
>>> import neopixel
>>> help(neopixel)

object <module 'neopixel' from 'neopixel.pu'> is of type module
  __file__ -- neopixel.py
  __version__ -- 0.1.0
  NeoPixel -- <class 'NeoPixel'>
  bitstream -- <function>
```

bitstream() is the same as the function in *machine()* (kind of redundant).

NeoPixel allows you to initialize an I/O pin for output to a NeoPixel as:

```
np = neopixel.NeoPixel(pin, n, type, timing)
```

where

- pin = pin that's connected to the neopixel
- n = number of neopixels
- type = 3 for RGB LEDs, 4 for RGBW LEDs
- timing = 1 for 800kHz, 0 for 400kHz (most are 800kHz)

For example, to drive eight RGB neopixels connected to pin #12, you would use

```
np = neopixel.NeoPixel(12, 8, bpp=3, timing=1)
```

NeoPixel contains several sub-options:

```
>>> import neopixel
>>> help(neopixel.NeoPixel)

object <class 'NeoPixel'> is of type type
  fill -- <function fill at 0x20012540>
  write -- <function write at 0x20012550>
  ORDER -- (1, 0, 2, 3)
```

- fill(): make all NeoPixels the same color
- write(): Update the NeoPixel (uses the *bitstream* command
- ORDER: changes the order to red-green-blue when using the *neopixel* library

Note: There is an error in the NeoPixel library. The timing used is

- [400, 850, 800, 450]

which works for most NeoPixel strips, but it does not work for the NeoPixel on the 52Pi board. If you want to drive the NeoPixel on the 52 pi board, you need to use the *bitstream()* command from before.

Example:

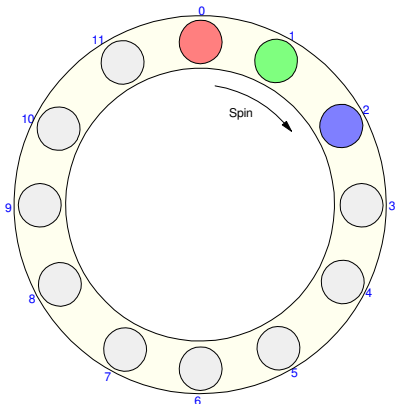
- Initialize a 12-element neopixel ring
- Set the default color to (0,0,0)
- Set the color of the first three LEDs to red - green - blue

```
import machine, neopixel

p = machine.Pin(12)
np = neopixel.NeoPixel(p, 12, bpp=3, timing=1)

np.fill([0,0,0])
np[0] = (50,0,0)   # red
np[1] = (0,50,0)   # green
np[2] = (0,0,50)   # blue
np.write()
```

Driving three NeoPixels using the *neopixel* library.
Note: the NeoPixel on the 52Pi board color isn't correct due to the timing of the neopixel library being off

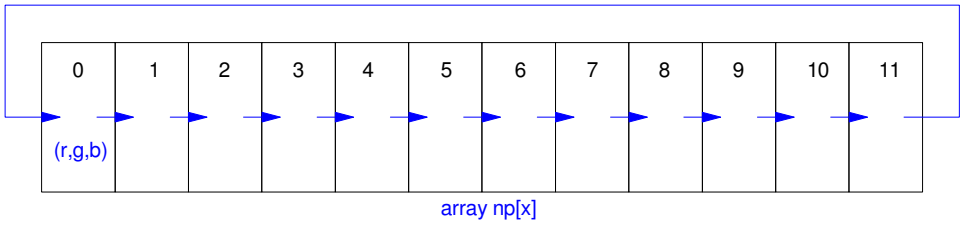
Fun with NeoPixels

LED Race: Turn on three LEDs as red / green / blue. Have those three LEDs circle around and around. Move the LEDs to pin #11 to avoid the onboard NeoPixel



Turn on three LEDs and make them spin around the ring

There are several ways to do this. In the following program, the array *np[]* is used as a circular stack. Each loop, the elements are pushed up one with *np[11]* shifting into *np[0]*



In order to make the LEDs rotate, array np[] is treated like a circular stack

```
import machine, neopixel, time

N = 12
p = machine.Pin(11)
np = neopixel.NeoPixel(p, N, bpp=3, timing=1)

n = 0
np.fill([0,0,0])
np[0] = (50,0,0)     # red
np[1] = (0,50,0)     # green
np[2] = (0,0,50)     # blue

while(1):
    temp = np[11]
    for i in range(0,10):
        np[11-i] = np[10-i]
    np[0] = temp
    np.write()
    time.slep(0.1)
```

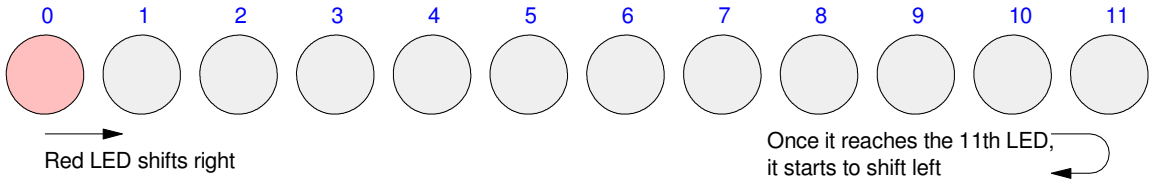
Program to make a red / green / blue light spin around a 12-element NeoPixel array attached to pin #11

LED Bounce:

This program makes the LED bounce back and forth between the first and last LED.

- n counts up to 11
- Once n reaches 11, it counts down to zero
- It then repeats over and over

All LEDs are off except for np[n] which is red:



LED Bounce: A red LED bounces back and forth

```
import machine, neopixel, time

N = 12
p = machine.Pin(11)
np = neopixel.NeoPixel(p, N, bpp=3, timing=1)

n = 0
dn = 1

while(1):
    if(n == 11):
        dn = -1
    if(n == 0):
        dn = +1
    n += dn
    np.fill([0,0,0])
    np[n] = (50,0,0)
    np.write()
    time.slep(0.1)
```

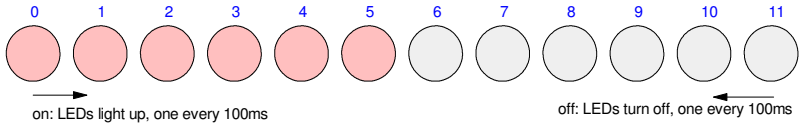
Program to make a single red LED bounce back and forth

Light Saber:

For Star Wars fans, turn on and off a light saber:

- Button GP15 turns on the light sabre
- Button GP14 turns off the light sabre.
- When turned on, the LEDs start lighting up from 0 to 11 every 100ms
- When turned off, the LEDs start turning off from 11 to 0 every 100ms

This is controlled by variable n, which counts up to 11 or down to 0 when turned on or off.



Light Saber: LEDs turn on or off when you press buttons GP15 (on) and GP14 (off)

```
from machine import Pin
from time import sleep
from neopixel import NeoPixel

N = 12
p = Pin(11)
np = NeoPixel(p, N, bpp=3, timing=1)
p_on = Pin(15, Pin.IN, Pin.PULL_UP)
p_off = Pin(14, Pin.IN, Pin.PULL_UP)

n = 0
power_on = 0

while(1):
    if(p_on.value() == 0):
        power_on = 1
    if(p_off.value() == 0):
        power_on = 0
    if(power_on == 1):
        n = min(n+1, N)
        np[n-1] = (50,0,0)
    else:
        n = max(n-1, 0)
        np[n] = (0,0,0)
    np.write()
    time.slep(0.1)
```

Program to turn your NeoPixel array into a light sabre (sound effects not included)

Summary

NeoPixels aren't too hard to use with a Raspberry Pi Pico and Python.

- You can achieve slightly more accurate timing using the *bitstream()* command along with binary files. To drive N LEDs, you need a binary file with 3N bytes in the order of green / red / blue.
- You can also drive NeoPixels using the NeoPixel library. In this case, to drive N neopixels, you set up an array of N tupples, with the order of the lights being (red, green, blue).

References

- https://www.hackster.io/Infineon_Team/controlling-neopisels-with-micropython-1ca0d6
- https://rancomnerdtutorials.com/micropython-ws2812b-addressable-rgb-leds-neopixel-esp32-3sp8266/