NeoPixels

ECE 476 Advanced Embedded Systems

Jake Glower - Lecture #26

Please visit Bison Academy for corresponding
lecture notes, homework sets, and solutions

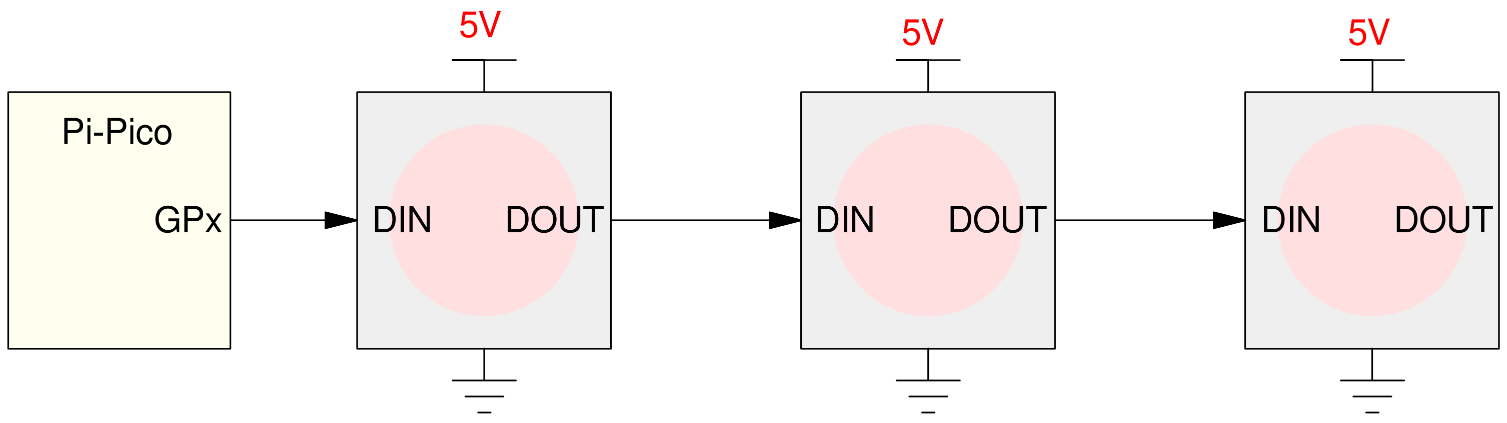
Introduction:

NeoPixels are RGB LEDs with a single-wire interface.

- aka WS2811 or WS812

Several NeoPixels can be cascaded

- Connect DOUT to DIN of the next NeoPixel:



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

Driving a NeoPixel String

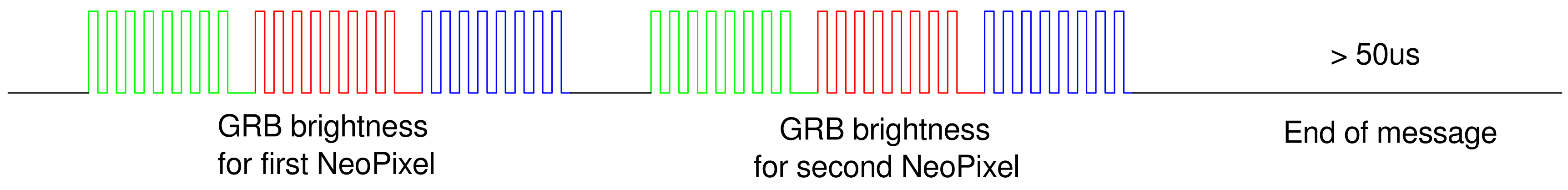
Send a series of 24 bits for each NeoPixel

- Three bytes for green - red - blue
- 255 = 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.

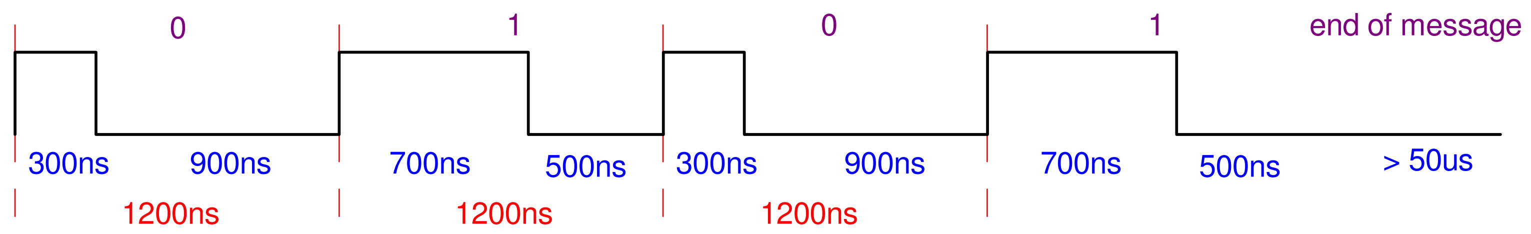
An idle time of >50us indicates end of message


> 50us
GRB brightness for first NeoPixel
GRB brightness for second NeoPixel
End of message

Sending 1's and 0's

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

bitstream()

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

Driving One NeoPixel:

Drive the NeoPixel on your board

- Red
- pause one second
- Green
- pause one second
- Blue
- pause one second
- repeat

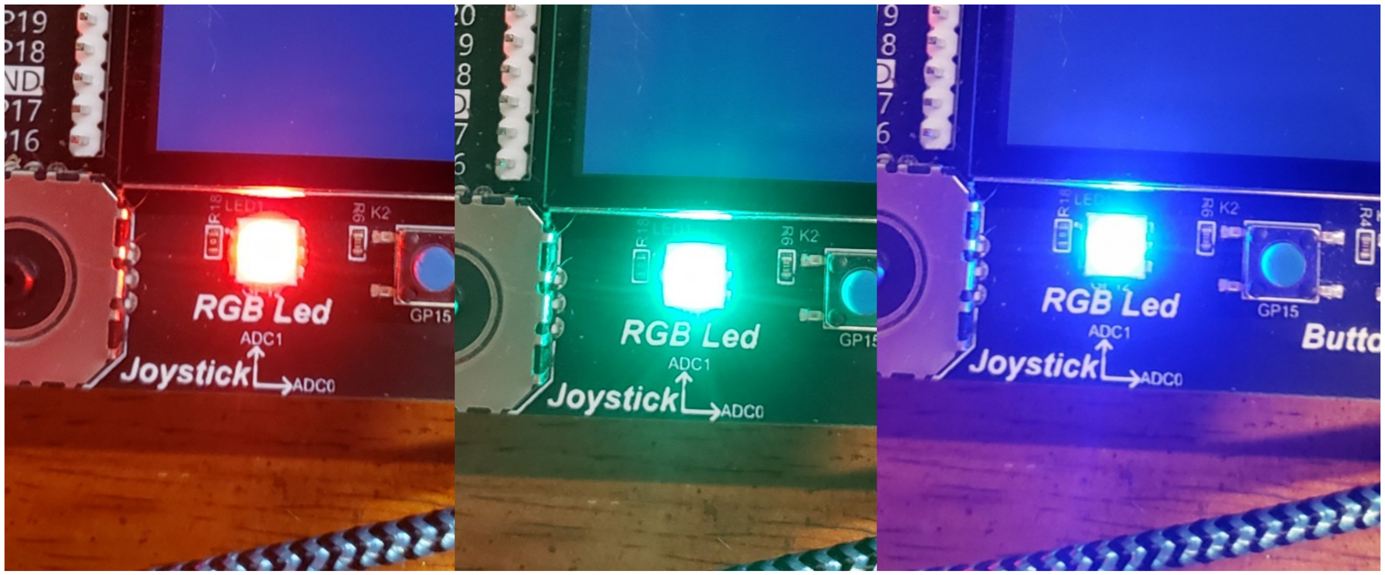
```
from machine import Pin, bitstream
from time import sleep

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')
```

Result

NeoPixel changes from

- Red - Green - Blue



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

Driving 12 NeoPixels

Attached to pin #12

- Same as NeoPixel on 52Pi board

Set up a byte-array

- 3N elements (36)
- Order = green / red / blue

Once set up, drive the NeoPixel using

bitstream()

```
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)
```

Result

NeoPixel[0] = 10 / 20 / 30

- bright light
- also show up on 52Pi board

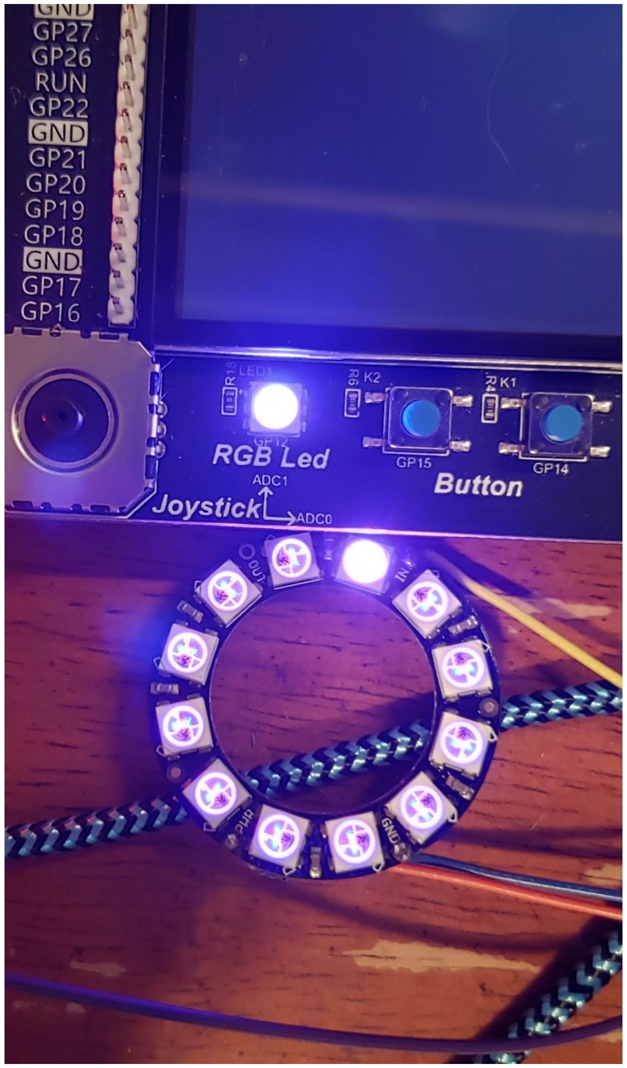
NeoPixel[1..11] = 1 / 2 /3

- dim lights

Note: X is a byte-array

- order is green / red / blue
- for 12 LEDs

```
>>> print(X)
bytearray(b'\x0a\x14\x38\x01\x02\
x03\x01\x02\x03\x01\x02\x03\x01\x
02\x03\x01\x02\x03\x01\x02\x03\x0
1\x02\x03\x01\x02\x03\x01\x02\x03
\x01\x02\x03\x01\x02\x03')
```



Changing One LED

X is an array of 36 bytes

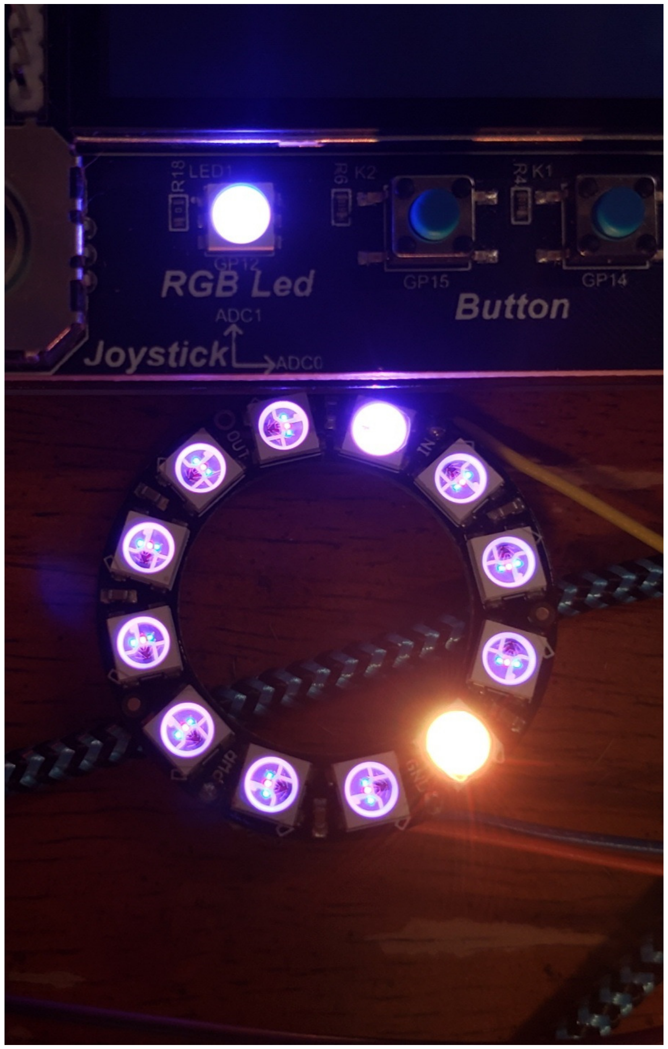
- 3 x 12

To change the color of LED N

- X[3N] = green
- X[3N+1] = red
- X[3N+2] = blue

Example: Make LED #4 orange

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

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



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

Initializing using the NeoPixel Library

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)

Example: Eight RGB neopixels connected to pin #12

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

neopixel.NeoPixel

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

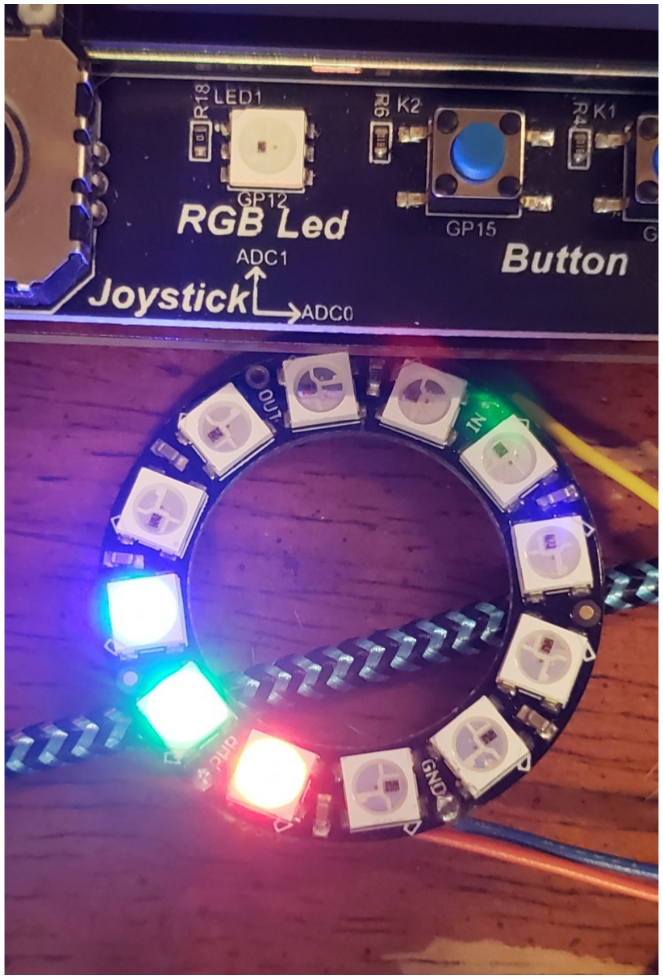
- *[400, 850, 800, 450]*

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.

RGB 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

```
from machine import Pin
from neopixel import NeoPixel

p = Pin(12)
np = 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()
```



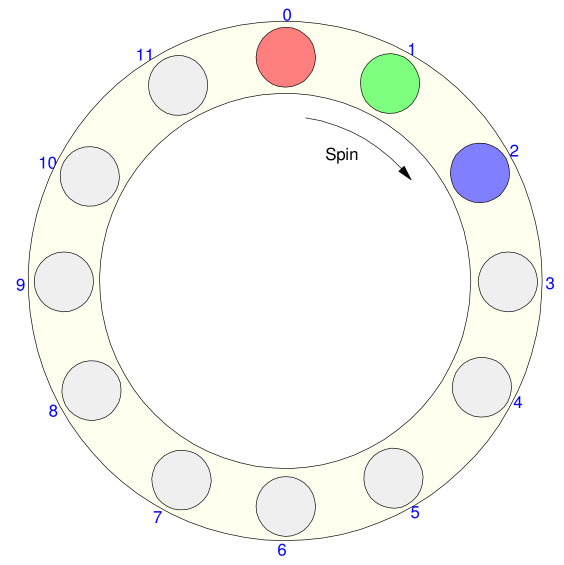
LED Race

Turn on three LEDs

- Red / Green / Blue

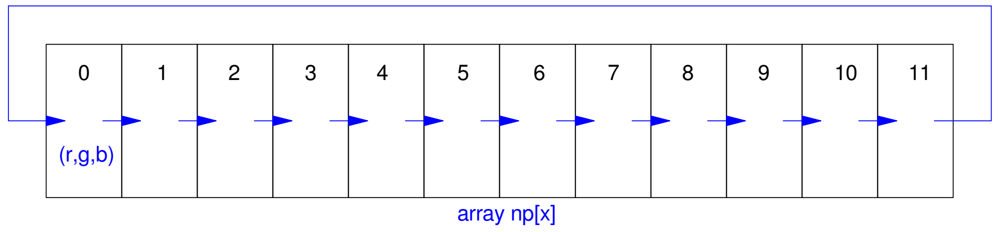
Rotate these around a 12-element ring

- One shift every 100ms



Strategy:

- Shift the elements of *np[]* every 100ms
- Treat *np[]* as a shift register
  - 0 goes to 1
  - 1 goes to 2
  - 11 goes to 0



LED Race: Code

Connect the ring to pin #11

- Avoids 52Pi LED on pin #12

Start with all LEDs off

- np.fill([0,0,0])

Set first three LEDs to r/g/b

Then

- Shift the data one slot
- Every 100ms

there are other ways to do this...

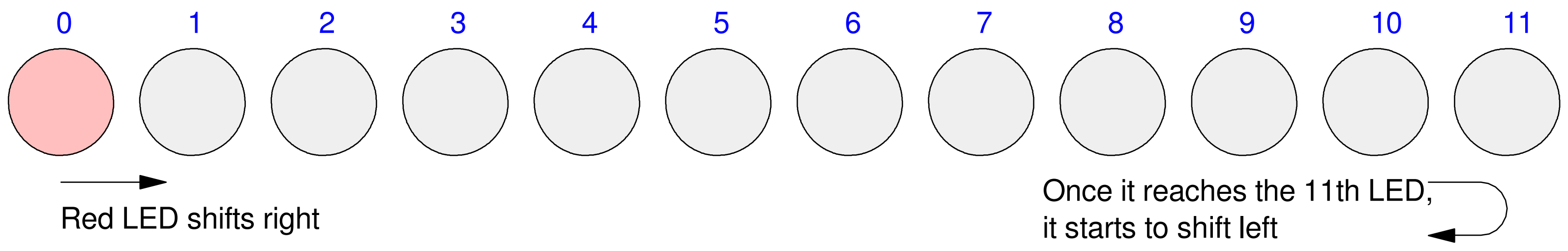
```
from machine import Pin
from neopixel import NeoPixel
from time import sleep

N = 12
p = Pin(11)
np = 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()
    sleep(0.1)
```

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 Code

Different approach

- Use a pointer (n)
  - n counts up to 11
  - then counts down to 0
- One count every 100ms

Element n is red

- All the rest are off

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

N = 12
p = Pin(11)
np = 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()
    sleep(0.1)
```

Light Saber:

sound effects not included

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

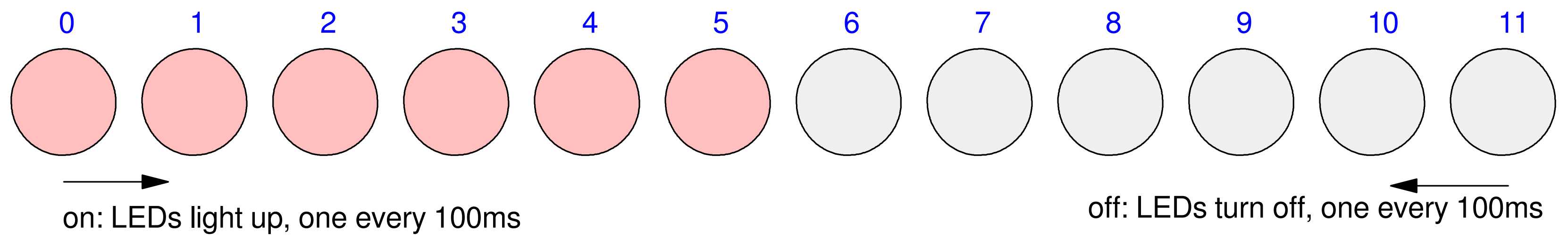
- Button GP15 turns on the light sabre
- Button GP14 turns off the light sabre.

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


0
1
2
3
4
5
6
7
8
9
10
11
on: LEDs light up, one every 100ms
off: LEDs turn off, one every 100ms

Light Saber Code

power_on

- 1 when saber is turned on
  - button GP15
- 0 when turned off
  - button GP14

np

- Array of 12 tupples
- Color of each LED

n

- pointer
- Which light is on / off
- Count up when turned on
- Count down when turned off
- One count every 100ms

```
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.sleep(0.1)
```

Summary

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

You can use *bitstream()*

- Uses a binary array of 3N bytes
- Order = green / red / blue
- Drives the LED on the 52Pi board
- Drives LED arrays

You can use the library *neopixel*

- Uses an array of N tupples
- Order = (red, green, blue)
- Doesn't work on 52Pi's LED
  - timing is off
- Does work with LED arrays

References

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