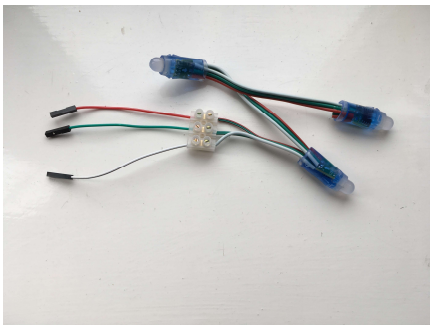
Raspberry Pi Python Individually Controllable LEDs (ws2811)

Tutorial by Jonathan Teague - Public Domain
2nd March Jan 2019 - www.cotswoldjam.org

This tutorial will cover using individually controllable strings of LEDs based on the ws2811 controller also known as 'NeoPixels'.

The Kit



In your bag you will find the following components:

- 3 x M-F Jumper leads (pin to socket)
- 1 x 3-way 3A connector strip
- 1 x String of 3 5V ws2811 LEDs

Putting it together:

WARNING: DO NOT TURN ON THE RASPBERRY Pi UNTIL YOU HAVE HAD YOUR CONNECTIONS CHECKED BY A TUTOR!!

Normally we get you to put the kit together as that's all part of the fun but for once there is no assembly, we have already screwed the LED string and the jumper leads into the connector block so all you have to do is plug the 3 leads to the Pi.

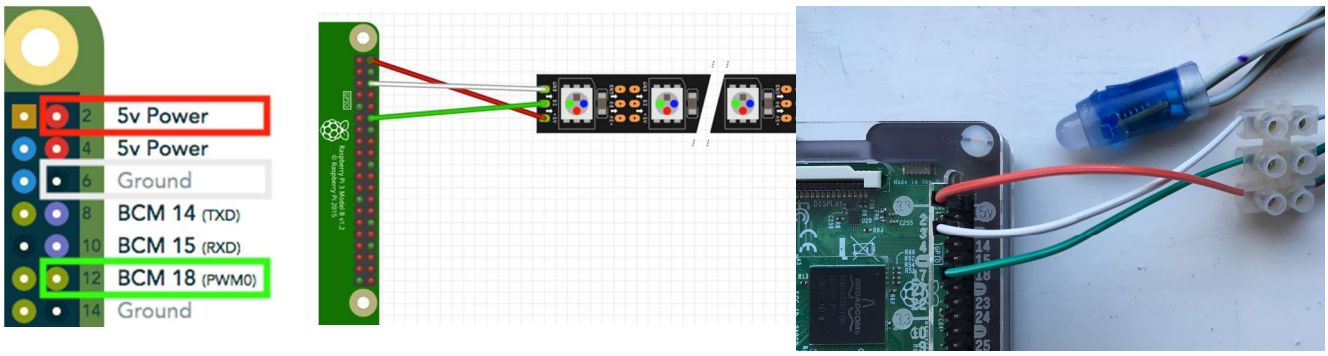
The colour of the jumper wires from the connector block is not important – it is the colour of the wires that come out of the connector block to the string that is important.

Step 0: Make sure your Pi is shut down and the power lead disconnected.

Step 1: Take the jumper lead that goes to the WHITE wire to the LEDs and plug the socket (female end) into GND(Pin6) on the Pi.

Step 2: Take the jumper lead that goes to the RED wire to the LEDs and plug the socket (female end) into 5V(Pin2) on the Pi.

Step 3: Take the jumper lead that goes to the GREEN wire to the LEDs and plug the socket (female end) into GPIO18(Pin12) on the Pi.



Now, no matter how confident you are that you've got it all right, call over a tutor and get your wiring checked before you power up the Pi. Once you're given the OK connect the power lead to the Pi and wait for it to boot.

The program

Power up your Raspberry Pi. From the desktop menu, select Programming - Python 3 (IDLE). Then use File, New File to create a new program.

Type in the following program then use File, Save to save your program as the name of your choice (don't forget to put `.py` on the end).

```
#!/usr/bin/python3
import board
import neopixel
import time

pixels = neopixel.NeoPixel(board.D18, 3, auto_write=False)

pixels[0] = (255, 0, 0)
pixels[1] = (0, 255, 0)
pixels[2] = (0, 0, 255)
pixels.show()

time.sleep(2)

pixels.fill((0, 0, 0))
pixels.show()
```

Running your program.

You can't run your program from within IDLE ☹. This is because the libraries need elevated permissions to access the hardware on the Pi.

To run the program start a Terminal – click the icon on the top then once the terminal window opens type:

```
sudo python3 thenameyouchose.py
```

If it's worked you should see the 3 LEDs light up red, green and blue for two seconds and then turn off.

What does the program do?

The imports load the NeoPixels libraries (and time so we can insert a delay).

```
pixels = neopixel.NeoPixel(board.D18, 3, auto_write=False)
```

This tells the library we've connected our LED string to GPIO 18, there are 3 LEDs and `auto_write=False` means only update the LEDs when we call `pixels.show()`

```
pixels[0] = (255, 0, 0)
```

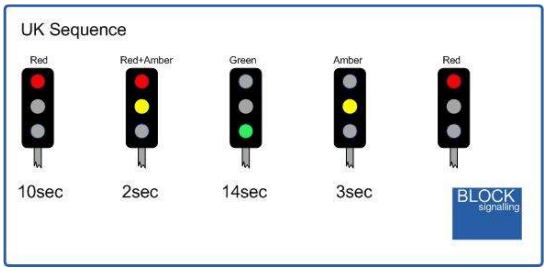
This sets the first LED to GREEN, the 3 values are (G, R, B) - so each of the 3 LEDs lights up a different colour.

```
pixels.fill((0, 0, 0))
```

This sets all the pixels to 0 which means off, if we don't do this the LEDs will stay on – that's because the ws8211 IC keeps doing what you last told it until you tell it something different.

What next?

Make your program show a traffic light sequence.



Hint – put it in a `while True:` block so it goes on forever.

> And don't forget to indent your code within the `while` – like this line is!

Now do your own thing :-

If you want you can copy the sample program in **`~/python/addrleds/neopixels_jam.py`** with:

```
cp /python/addreleds/neopixels_jam.py .
```

Open it in IDLE, work out what it does, start customising it.

To run the program type:

```
sudo python3 neopixels_jam.py
```

How does it work?

Each LED is connected to a ws2811 IC – it's the IC that's called a ws2811, the LED is just a tri-colour LED. The Pi generates a long sequence of 0's and 1's. 24 bits for each LED (3 sets of 8-bits for each colour Red|Green|Blue), so for our string of 3 LED the Pi will generate 3 x 24 bits = 72 bits.

The first device in the string takes the fist 24 bits and uses those to set the Red|Green|Blue on its LED. It then passes the remaining 48 bits on to the second LED. That in turn takes what are now the first 24 bits from what it has received as its RGB and passes on the remaning 24 bits to the last LED.

For full details the datasheet is here: http://www.world-semi.com/DownLoadFile/129

Making it work at home

Unlike most Cotswold Jam tutorials you'll have to do some extra steps to get this one to work at home. The libraries for the software drivers for the LEDs are not included in the standard NOOBS or Raspbian images you get from raspberrypi.org

To get NeoPixels working follow the instructions here:

https://learn.adafruit.com/neopixels-on-raspberry-pi

Notes on using ws2811 in the real world

We've only given you a short string of LEDs as that's all the Pi can reliably power. If you're using a long strip of ws2811 or ws2812 then there are two things you'll need to consider.

1. ws2811 strips can be 5V or 12V, ws2812 are 5V. The Pi can only provide 5V at a low current sufficient to drive a few LEDs. To drive long strips at 5V or any at 12V you'll need to provide additional power, possibly at multiple points along the strip. If you're going to run a strip all round your bedroom ceiling ☺ then the Pi is only going to provide the control logic for the LEDs not the power.
2. The control logic for ws2811 & ws2812 is specified at 5V, the Pi GPIO pins are 3.3V. For this tutorial we get away with using GPIO but to be sure it's going to work you would probably want to use a logic level shifter to turn the Pi's 3.3V into 5V.