AdaFruit NeoPixels

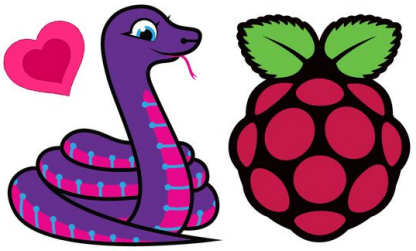
AdaFruit NeoPixels are small multi-colour (RGB) LEDs that can be chained together and be individually addressed by a microcontroller to create many interesting and sparkly installation or wearable tech projects! We will be connecting NeoPixels to the Raspberry Pi computer today and using the CircuitPython library from AdaFruit to control them.

Theoretically you can connect as many NeoPixels together as you want; however, the more NeoPixels you connect, the greater the amount of power they demand. When powering them directly from the Raspberry Pi (as we will do today) you are able to power a few at a time.

NeoPixels also operate on a 5V signal whereas the Pi operates on a 3.3V signal. Again, for today's purpose, we are going to use the 3.3V signal from the Pi to control the NeoPixels. If you were building a large scale project, it is best to use a level shifter that steps the control signal from the Pi up to 5V.

This website has all the information you need:
https://learn.adafruit.com/neopixels-on-raspberry-pi?view=all

Install and enable libraries and interfaces



You do not need to carry out this section at the Jam as we have already completed this for you, but if you were to use one at home you would need to follow these steps.

1. First we check that PIP for Python 3 is installed:
   `sudo apt-get install python3-pip`
2. Now we enable I2C and SPI in menu "**5 Interfacing Options**" of raspi-config:
   `sudo raspi-config`
3. Reboot
4. Now check that both I2C and SPI are enabled:
   `ls /dev/i2c* /dev/spi*`
5. You should see this output:
   `/dev/i2c-1  /dev/spidev0.0  /dev/spidev0.1`
6. Now install the Raspberry Pi GPIO library:
   `sudo pip3 install RPI.GPIO`
7. Next install the AdaFruit Blinka library:
   `sudo pip3 install adafruit-blinka`
8. Finally install the AdaFruit CircuitPython NeoPixel libraries:
   `sudo pip3 install rpi_ws281x adafruit-circuitpython-neopixel`

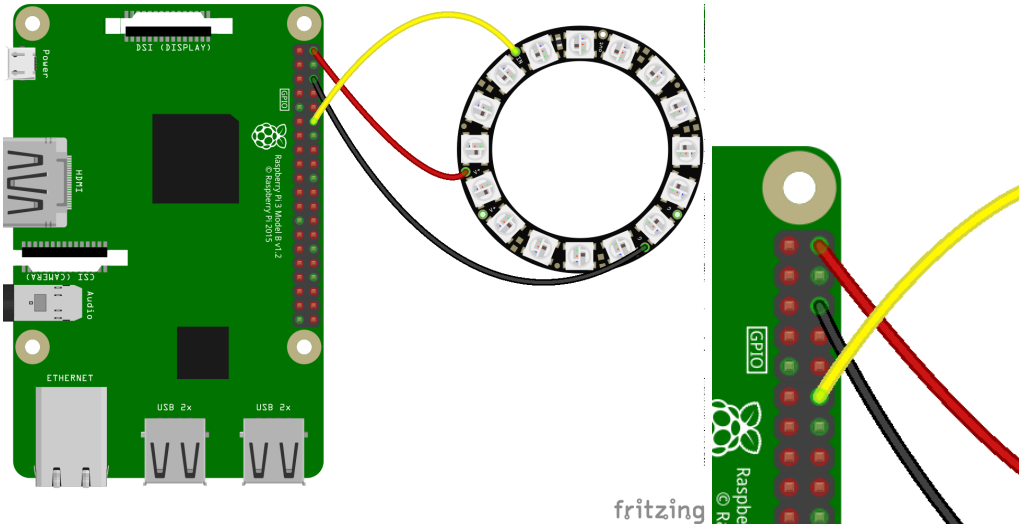
Wiring up our NeoPixels

Now we can connect our NeoPixels to our Raspberry Pi. It is best to do this with your computer switched off so that you can double check everything is connected correctly before powering it on.

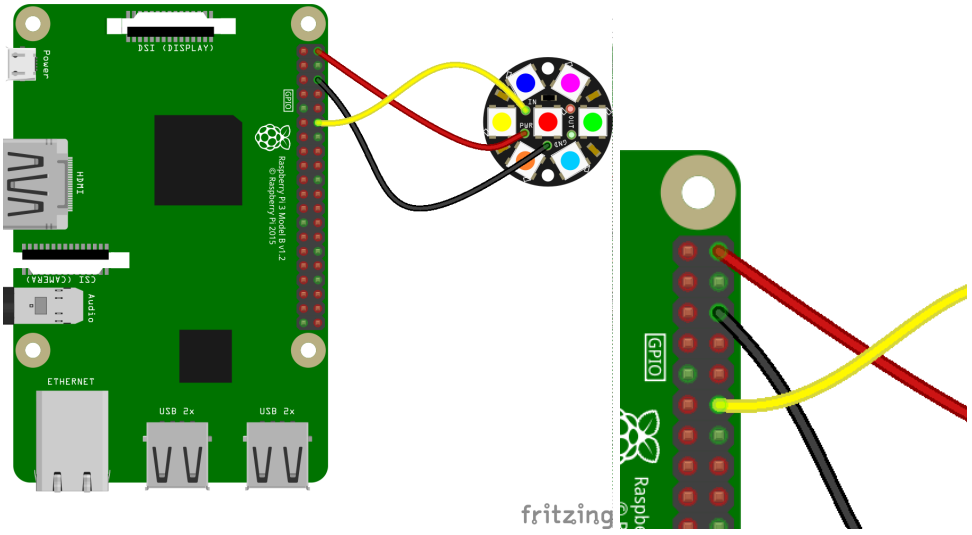
Use the diagram below to connect the type of NeoPixel you are using. The pin labelled “**V+**” connects to the 5 volt pin of the Pi (red cable). The pin labelled “**G**” or “**GND**” connects to a ground pin on the Pi (black cable). Finally the pin labelled “**In**” connects to GPIO18 on the Pi (yellow cable).

If you are using the NeoPixel Breadboard be sure that when you chain them together you are connecting the “**out**” from one NeoPixel to the “**in**” on the next.

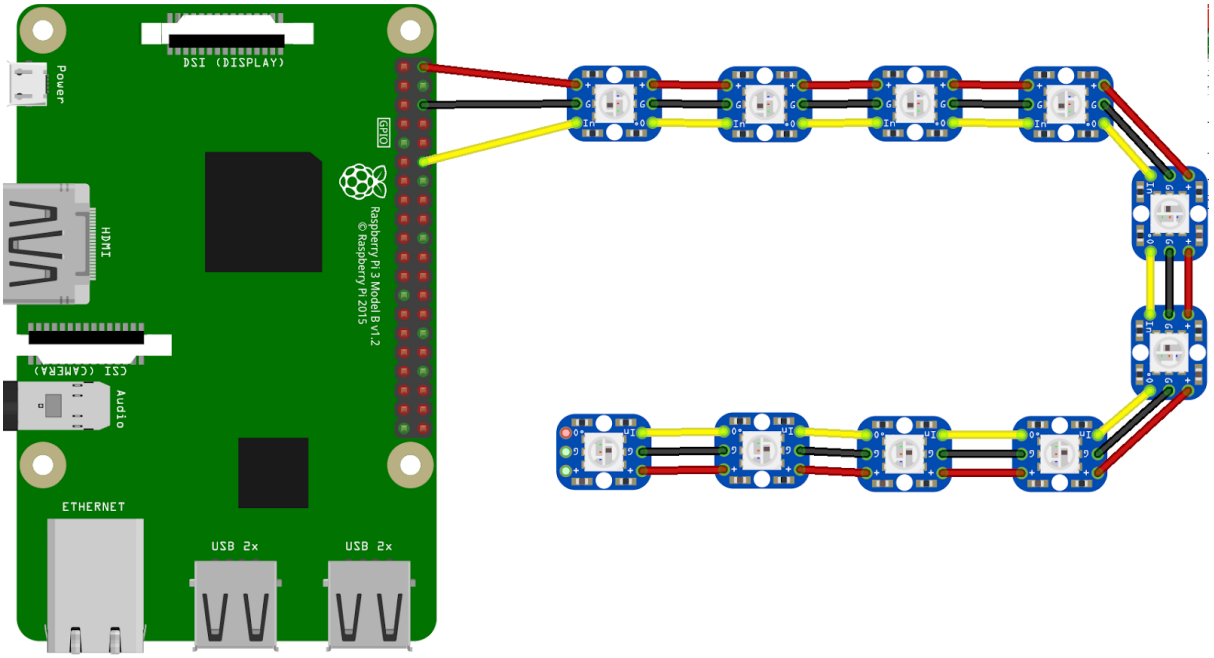
NeoPixel Ring



NeoPixel Jewel



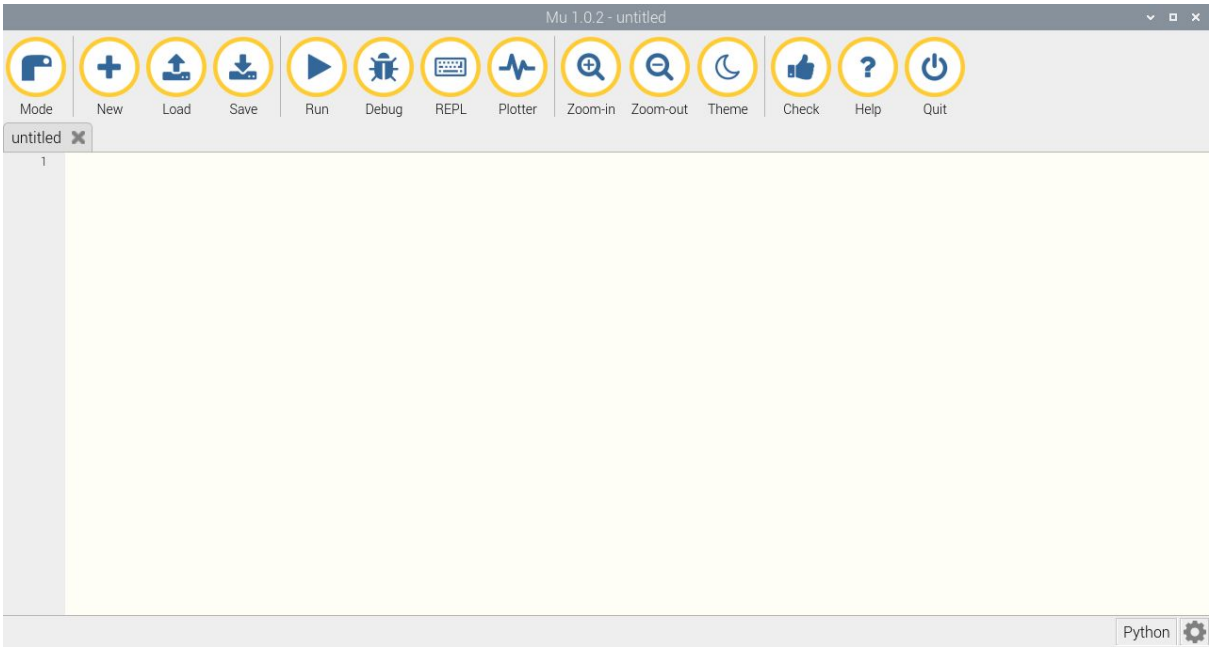
NeoPixel Breadboard



Writing our Python

Will will be using the Mu Python IDE. Open it by going to the "**Raspberry Pi menu > Programming > mu**". *If you do not have mu available on your Raspberry Pi, you can install by typing the following into a terminal window:*

```
sudo apt install mu-editor
```



Single Light

We will start by testing we can light up a single NeoPixel. Copy the following code into a new file in the mu editor. Change the value of the "**NUM_PIXELS**" constant to match the number of NeoPixels you have in your chain (16 for the Ring, 7 for the Jewel and however many you have chained together for the Breadboard). Save the file as "**single.py**" and run it in the terminal with this command:

```
sudo python3 /home/pi/mu_code/single.py
```

ngle.py

```
1  # import our python libraries
2  import board
3  import neopixel
4
5  # define our constant for the number of neopixels
6  # 16 for the Ring, 7 for the Jewel
7  NUM_PIXELS = 16
8  # define our constant for the GPIO pin number
9  PIXEL_PIN = board.D18
10 # The order of the pixel colors - RGB or GRB
11 # Some NeoPixels have red and green reversed!
12 ORDER = neopixel.GRB
13
14 # define our connection string
15 pixels = neopixel.NeoPixel(PIXEL_PIN, NUM_PIXELS, brightness=0.2,
16                            auto_write=False, pixel_order=ORDER)
17
18 # light the first pixel red
19 # colours are defined with a Red, Green and Blue tuple.
20 # 0 means none of that colour and 255 means all of that colour
21 pixels[0] = (255, 0, 0)
22 # we call pixels.show() to display the changes
23 pixels.show()
```

Light Them All!

Now we can create a new file with the following code. Save it as "**lightall.py**" and then run it in the terminal with this command:

```
sudo python3 /home/pi/mu_code/lightall.py
```

```
# import our python libraries
import board
import neopixel
import time

NUM_PIXELS = 16 # adjust to match
PIXEL_PIN = board.D18
ORDER = neopixel.GRB

# define our connection string
pixels = neopixel.NeoPixel(PIXEL_PIN, NUM_PIXELS, brightness=0.2,
                           auto_write=False, pixel_order=ORDER)

# light all the pixels red, sleep for 5 seconds
# then clear them all. note the double brackets here
pixels.fill((255, 0, 0))
pixels.show()
time.sleep(5)
pixels.fill((0, 0, 0))
pixels.show()
```

Wipe Pattern

We can now use a loop to go around each pixel in turn and create a wipe effect. Copy this code into a new file and save it a “**wipe.py**” and then run it in the terminal with this command:

```
sudo python3 /home/pi/mu_code/wipe.py
```

wipe.py

```
# import our python libraries
import board
import neopixel
import time

NUM_PIXELS = 16 # adjust to match
PIXEL_PIN = board.D18
ORDER = neopixel.GRB

# define our connection string
pixels = neopixel.NeoPixel(PIXEL_PIN, NUM_PIXELS, brightness=0.2,
                           auto_write=False, pixel_order=ORDER)

# loop forever then loop through number of pixels
# clear the all, set pixel at index position to
# red, sleep for 0.05 seconds, move to next pixel
while True:
    for i in range(NUM_PIXELS):
        pixels.fill((0, 0, 0))
        pixels.show()
        pixels[i] = (255, 0, 0)
        time.sleep(0.05)
        pixels.show()
```

As this code uses a “**while True**” loop it will run indefinitely. Press **Ctrl** and **C** together in the terminal to stop it running.

Rainbows!

This code uses two functions; “**wheel**” and “**rainbow_cycle**”. These functions are used to create a rather pretty rainbow effect! Copy this code into a new file, save it as “**rainbow.py**” and run it from the terminal with this command:

```
sudo python3 /home/pi/mu_code/rainbow.py
```

```python
# import our python libraries
import board
import neopixel
import time

NUM_PIXELS = 16 # adjust to match
PIXEL_PIN = board.D18
ORDER = neopixel.GRB

# define our connection string
pixels = neopixel.NeoPixel(PIXEL_PIN, NUM_PIXELS, brightness=0.2,
                           auto_write=False, pixel_order=ORDER)
```

```python
def wheel(pos):
    # Input a value 0 to 255 to get a color value.
    # The colours are a transition r - g - b - back to r.
    if pos < 0 or pos > 255:
        r = g = b = 0
    elif pos < 85:
        r = int(pos * 3)
        g = int(255 - pos*3)
        b = 0
    elif pos < 170:
        pos -= 85
        r = int(255 - pos*3)
        g = 0
        b = int(pos*3)
    else:
        pos -= 170
        r = 0
        g = int(pos*3)
        b = int(255 - pos*3)
    return (r, g, b) if ORDER == neopixel.RGB or ORDER == neopixel.GRB else (r, g, b, 0)
```

```python

def rainbow_cycle(wait):
    for j in range(255):
        for i in range(NUM_PIXELS):
            pixel_index = (i * 256 // NUM_PIXELS) + j
            pixels[i] = wheel(pixel_index & 255)
        pixels.show()
        time.sleep(wait)

while True:
    rainbow_cycle(0.001)
```

As this code uses a “**while True**” loop it will run indefinitely. Press **Ctrl** and **C** together in the terminal to stop it running.