University of Hull
Department of Computer Science
C4DI
Using Coloured Leds

Vsn. 1.0 Rob Miles 2014

LED Light Limitations

We've seen led (light emitting diode) lights in action ever since we wrote our first program. The Arduino output signals have enough power to light a single led. We can turn the led on or off, or we can use pulse width modulation (turning the power on and off very quickly) to simulate varying voltages and display different brightness levels.

However, this form of display is somewhat limited. We need a pin on the Arduino for every led that we want to control. What's worse, if we want to have different brightness levels on these leds the program in the Arduino has to work quite hard to turn the signals on and off and make the different analogue voltages.

What we really want is an led that is clever enough to be able to control its own brightness. The Arduino would just tell it how bright the led was to be and then the led would shine at that level. To make things even nicer we could perhaps ask for three leds (one red, one green and one blue) in the package so that we could set any colour we like into the led. And, just to round this off it would be nice to have a way we could put the lights into chains so that a single Arduino could control a very large number of them from a single pin. Turns out that this has been done for us, in the form of the Neopixel or WS2812 Integrated Light Source device.

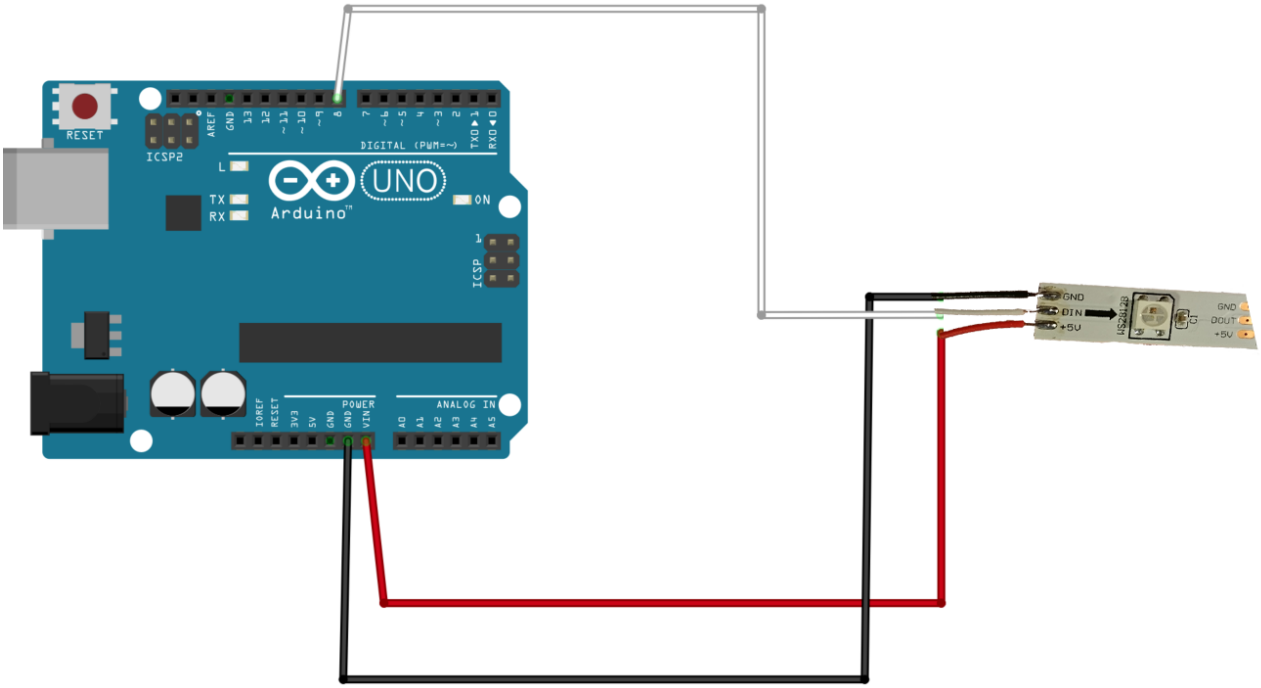
The NeoPixel LED

A single NeoPixel device is absolutely tiny, as you can see. They can be used in a daisy chain arrangement, with the output of one linked to the input of the next. A single Arduino pin can be used to control a very large number of lights in this way. Software in the Arduino waggles the data line up and down to signal the amount of red, green and blue the led is to display. When one led has acquired its settings it then passes the signal onto the next one in the chain, and so on.

You can buy NeoPixels in a variety of configurations including panels of 64 leds, or as strips or rings. They are a great way to add a bit of impressive lighting effects to your projects. And they are very bright.

Each NeoPixel will consume around 60 milliamps if you fully light the red, green and blue elements. I've found that you can run rings of up to 16 or so on an Arduino connected to the usual computer USB power supply, but if you want more lights you will have to find an extra power supply.

Connecting NeoPixels to an Arduino



We are going to use a tiny strip of NeoPixels with three lights on it. If you like you can go shopping and buy a much larger one.

You can connect the strip to the Arduino in the manner shown above. The strip needs power and ground connections (you can use the ones on the Arduino for up to 16 or so leds) and the data connection can be any of the digital pins on the Arduino. We can configure this in software.

Connect the strip of lights to your Arduino.

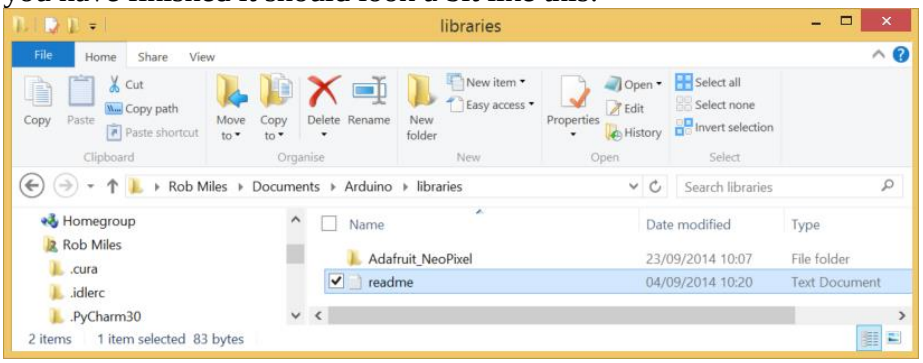
Installing the NeoPixel Library

We are going to use some software provided by AdaFruit, one of the companies that sell NeoPixels. We need to load this library into our Arduino installation before we can use it.

1. Go to the library page https://github.com/adafruit/Adafruit_NeoPixel
2. Press the Download Zip 

button to download the archive containing the libraries. Save the file in a sensible location.
3. You should have a file called `Adafruit_NeoPixel-master.zip`. Double click it to open it.
4. Inside the zip archive you will find a folder called `Adafruit_NeoPixel-master`. You need to copy this into the libraries folder in your Arduino installation. You should find an `Arduino` folder in your documents folder. This should contain a `libraries` folder, which is where we want to put the NeoPixel library.
5. Drag the `Adafruit_NeoPixel-master` library from the zip archive into the libraries folder. Then rename the folder to `Adafruit_NeoPixel`. When

you have finished it should look a bit like this:



Now we can use the NeoPixel libraries in our Arduino programs. You only have to copy the library files once. They can be used in any program that you create.

Turning on Lights

Now we can use the library to make our lights work.

```
#include <Adafruit_NeoPixel.h>

#define PIN 8

Adafruit_NeoPixel strip = Adafruit_NeoPixel(3, PIN, NEO_GRB + NEO_KHZ800);

void setup()
{
  strip.begin();
  strip.setPixelColor(0, 255, 0, 0 );
  strip.setPixelColor(1, 0, 255, 0);
  strip.setPixelColor(2, 0, 0, 255);
  strip.show();
}

void loop()
{
}
```

This is a simple program that sets the colour of each led. The `strip` variable is created and told the number of leds in the strip (in our case 3) and the Arduino pin that the strip is connected to (in our case pin 8). We can then use the `setPixelColor` method to set a colour for each led on the strip. The `setPixelColor` method (note the American spelling) is given the number of the led to write to, followed by the amount of red, green and blue to be displayed. The maximum value is 255 (as bright as possible). If you set a value 0 this colour is not displayed.

Note that the display will not actually change until you call the `show` method.

1. Load the program into your Arduino and run it. The first led should be red, followed by green and blue.
2. Change the program to make all the leds turn blue.
3. Try and make a traffic light display, red at the top, yellow in the middle and green at the bottom. You might need to do some research to find out how to make the yellow colour.

Using Randomness to Make the Lights Twinkle

If we want to make something a bit more interesting we can set the lights to random colours. The loop method in the previous sketch does nothing, so the display doesn't change.

```
void loop()
{
  byte red = random(0,256);
  byte green = random(0,256);
  byte blue = random(0,256);
  strip.setPixelColor(0,red,green,blue);
  strip.show();
}
```

This loop method picks random values for red, green and blue and then sets the colour of the first led to the random colour. Because the loop method is repeatedly called when the Arduino program is running this should result in a flickering display.

1. Add the loop method above to your program and note what happens.
2. Is this any good?
3. Why not?

The twinkling doesn't work very well because it is happening so quickly that we can't see the individual colours. We need a way of slowing things down so that we can see the lights. The delay method can be used to make the program pause for a particular number of milliseconds (thousandths of a second).

```
delay(300);
```

1. Add the call of delay to the loop method and run the program again. You should now have a single twinkling star.
2. Improve the program so that the program picks a different random colour for each of the lights.

Proper Twinkling Stars

Real twinkling stars don't all twinkle at the same time. Otherwise we would call them "flashing" stars. If we want to make something twinkle we have to add a bit more code.

```
void randomiseLed(byte ledNumber)
{
  byte red = random(0,256);
  byte green = random(0,256);
  byte blue = random(0,256);
  strip.setPixelColor(ledNumber,red,green,blue);
  strip.show();
}
```

I've started by adding a method called `randomiseLed`. This picks a random colour and then sets a particular led to that colour. The number of the led to be set is passed as a parameter and is used in the call of setPixelColor to determine which led is to be changed.
Once we have this method in place we can use it to control any of the leds:

```
void loop()
{
  int control = random(0,500);
  if(control == 0) randomiseLed(0);
  if(control == 1) randomiseLed(1);
  if(control == 2) randomiseLed(2);
  delay(1);
}
```

This loop method picks a random number and then sets the colour of one of the leds if the value matches one of them. The method also contains a delay statement that will cause it to wait 1 millisecond each time it runs.

The random number being generated is in the range 0 to 499. Each value has a 1 in 500 chance of being generated. If the loop method is running 1,000 times a second (which it is because of the delay) we should see the lights change once every half second.

1. Add the `randomiseLed` method and change the `loop` method to the one above.
2. Run the program and see if you like the flickering effect. How would you make the lights change more rapidly?

At this point you are on your own, and I invite you to experiment with different colour combinations.

Rob Miles 2014