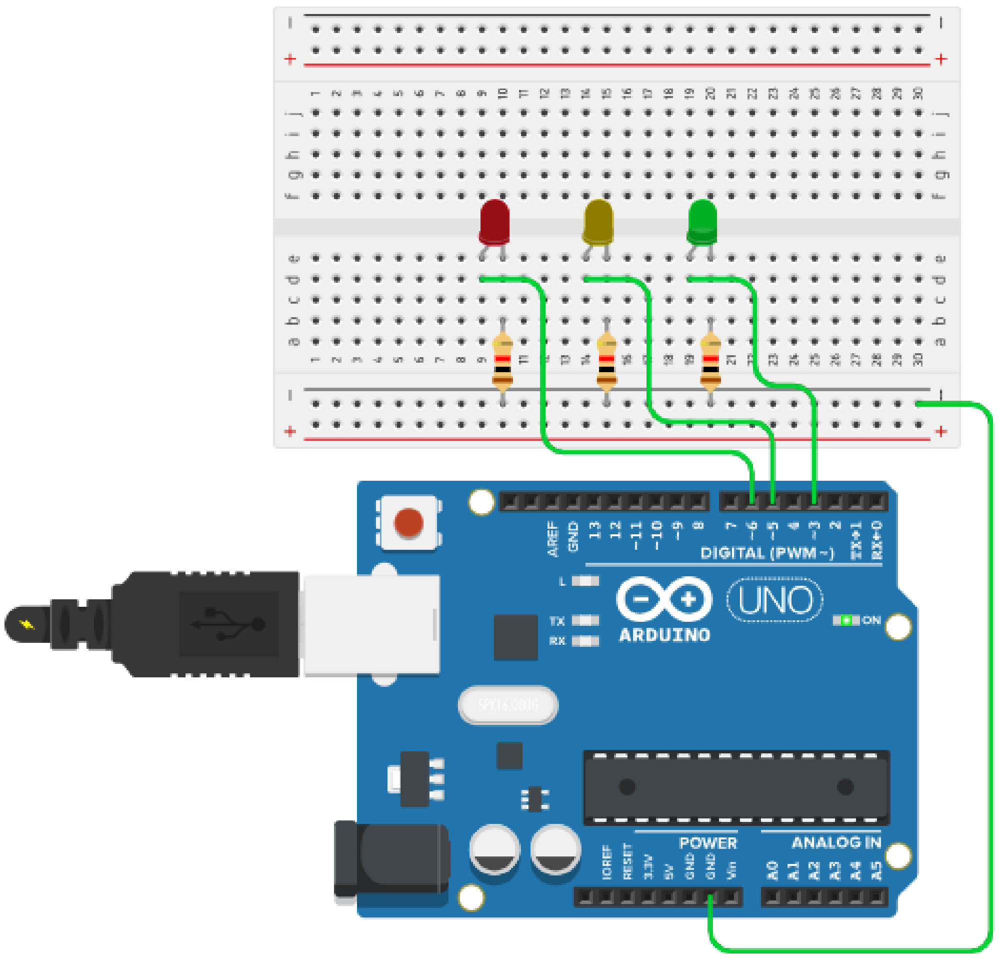
Activity: Write an Arduino Program to Control LED Brightness

Objective:



Since the invention of the first light bulb, there have been just 2 ways in which they have been used and are still the same. These two ways are:

a. Switching and
b. Dimming

Over the course of years, the type of technology behind light sources has seen changes such as incandescent, fluorescent and currently the LEDs are heading the revolution. But switching and dimming are still the only two ways in which electrical light sources are used in any form of application.

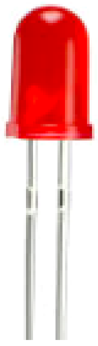
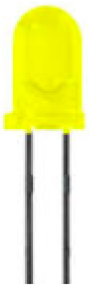
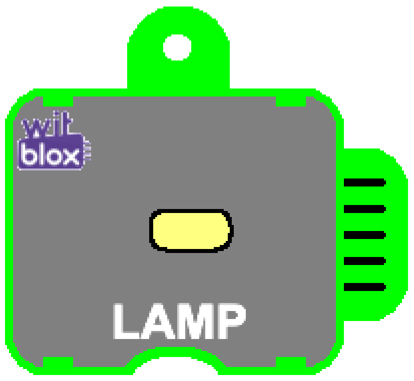
Switching simply means turning the light source On and Off, resulting in either full light intensity or no light at all. Dimming, on the other hand, means one can achieve a defined level of light intensity somewhere between no light and full light.

So far, in our journey, we have already seen, learnt and implemented Switching of LED lights. So, in today's activity we will address the other way light sources are used i.e. Dimming. This activity will be divided into 3 parts.

1. Controlling Brightness of an external LED.
2. Controlling Brightness of 3 external LEDs.
3. Controlling Brightness of the Lamp blox.

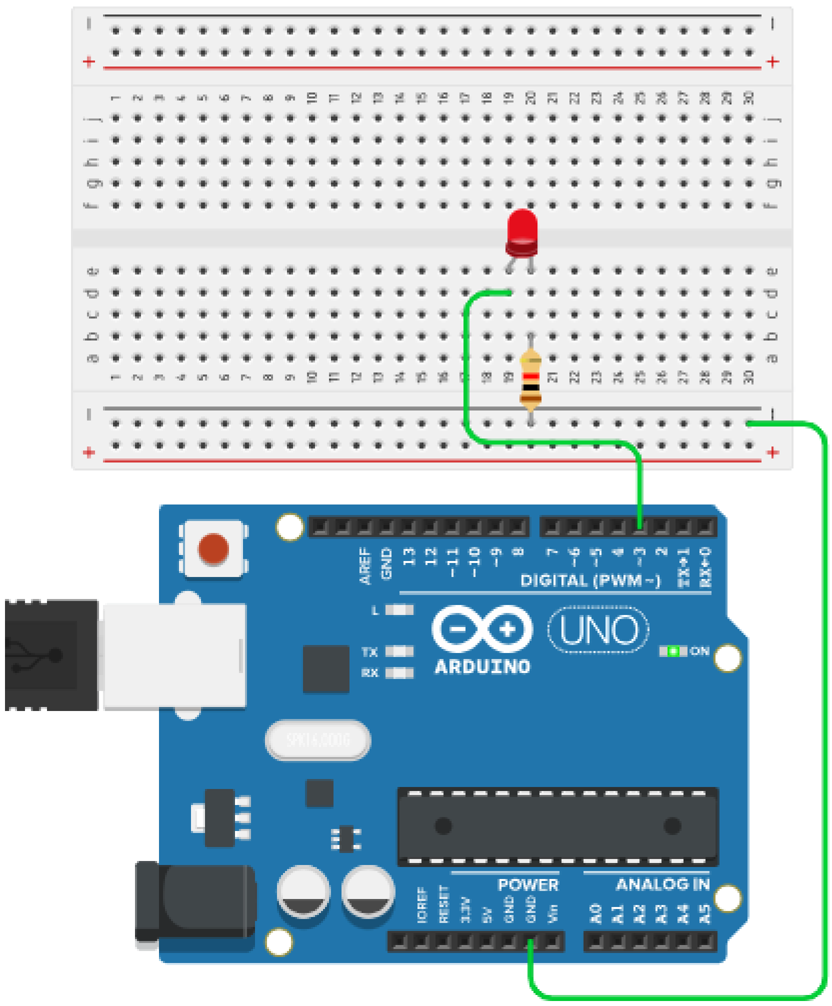
Materials Required:

S.no.	Name	Qty	Image
1	Arduino Nano	1	
2	Breadboard	1	
3	M - M jumper wires	10	
5	USB cable for Arduino Nano	1	

6	Resistors: 1k (Brown Black Red)	3	
7	Red LED	1	
8	Green LED	1	
9	Yellow LED	1	
10	Lamp blox	1	

Part 1:

Connection Diagram:



Explanation:

This connection diagram looks so familiar because it is the same as the first activity; which was switching an LED. but on close observation, you will notice that the Arduino pin, to which the LED is connected in this diagram, is not the same as the previous one.

In this connection diagram, the LED is connected to digital pin 3 (D3) of the Arduino Nano. Rest everything is exactly the same.

1. +ve pin of the LED is connected to Arduino digital Pin 3 (D3).
2. -ve pin of the LED is connected to one side of the 1k resistor.
3. The other side of the resistor is connected to the Gnd line of the breadboard.
4. Connect the Arduino’s 5v pin to the breadboard’s +ve line.
5. Connect the Arduino’s Gnd pin to the breadboard’s -ve line.

Arduino Code:

The statements used in this activity are slightly different from what we have been writing thus far. The basics are still the same. We use the “setup” function for things that need to be done just once. We use the “loop” function for things that need to be done repetitively forever (as long as powered on). And yet again we are going to use the “for” loop construct for repeating things a certain number of times.

Here is the Arduino code.

```
#define red 3               // defining pin 3 as "red"

void setup()
{
  pinMode(red, OUTPUT);     // declaring "red" pin as output
}

void loop()
{
  for(int i=0;i<255;i++)    // repeat for 0-255 diff levels
    {
        analogWrite(red,i);     //  write  pin  with  brightness
level
      delay(20);            // wait for sometime
    }
}
```

Explanation:

The only part that needs an explanation is the part that is controlling the brightness of the LED. Rest everything is already explained in detail in the previous activities. So, let’s just focus on the important part which are these statements.

```
  for(int i=0;i<255;i++)
    {
      analogWrite(red,i);
      delay(20);
    }
```

Instead of using a “digitalWrite” we are, here, using an “analogWrite”. The difference is, digitalWrite simply puts a High or Low voltage on the pin which, in turn, turns the LED On (with full brightness) or Off (with zero brightness) while analogWrite puts a voltage on the pin that is between the levels of full brightness and zero brightness.

And this makes the LED to glow with a light intensity that is proportional to the voltage available as a result of this analogWrite. Notice that the statement “analogWrite” does not mention a HIGH or LOW in it. Instead it mentions the pin number along with the level.

```
analogWrite(red,i);
delay(20);
```

Here, the pin is “red” (which is actually pin D3), the level is the variable “i” which ranges in value from 0 (zero brightness) to 255 (full brightness). The delay of 20 milliseconds is simply allowing us humans to actually be able to see the transition from minimum brightness to maximum brightness.

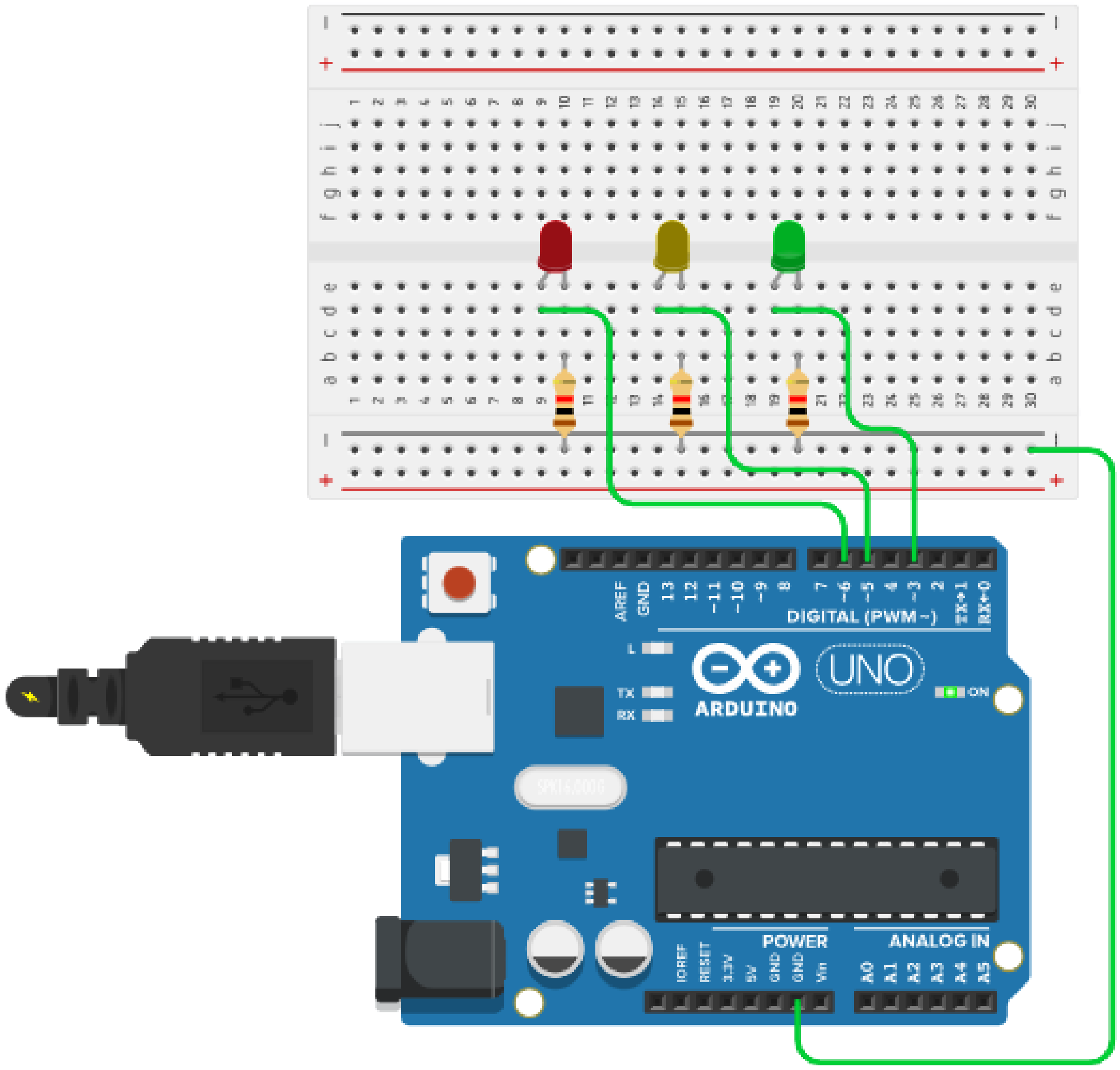
So, the analogWrite puts a voltage on the LED’s pin which keeps incrementing with every new value of “i” and every such increment is held for a 20 ms duration. You, the user, will see the LED’s brightness go from minimum to maximum in

255 x 20 = 5100 ms

That is 5.1 seconds. This is a fairly long time and so the transition in brightness is conveniently noticeable by an average person.

Part 2:

Connection Diagram:



Explanation:

In the second part of the activity, we use 3 external LEDs, each of a different color and each connected to a different pin on the Arduino. Take a look at the above connection diagram and you will observe the following:

1. The Green LED is connected to the Arduino's digital pin 3 (D3) such that the +ve pin of the LED is connected to D3 and the -ve pin of the LED is connected to a 1k (brown, black,red) resistor. While the other side of this resistor is connected to the Gnd line of the breadboard.

2. The Yellow LED is connected to the Arduino's digital pin 5 (D5) such that the +ve pin of the LED is connected to D5 and the -ve pin of the LED is connected

to a 1k (brown, black,red) resistor. While the other side of this resistor is connected to the Gnd line of the breadboard.

3. The Red LED is connected to the Arduino's digital pin 6 (D6) such that the +ve pin of the LED is connected to D6 and the -ve pin of the LED is connected to a 1k (brown, black,red) resistor. While the other side of this resistor is connected to the Gnd line of the breadboard.

4. The Gnd pin of the Arduino Nano is connected to the Gnd line of the breadboard and the 5v pin of the Arduino is connected to the +ve line of the breadboard.

We have already connected and controlled multiple LEDs before in our previous activity and this activity's connections look very similar as well. But, if you are a close observer, you will notice one thing immediately. And that is, instead of using the digital pins 2,3 and 4 like the previous activity we are using pins 3,5 and 6.

Why this strange way of selecting the pin numbers?

This is because the statement "analogWrite" works on only digital pins 3, 5, 6, 9, 10 and 11 of Arduino Nano (or UNO) as these are the only designated PWM (Pulse Width Modulation) output pins on the Arduino Nano.

We will dive deeper into the depths of this strange term PWM in the activities to come. For now, just remember that **digitalWrite is for Switching** and **analogWrite is for Dimming**.

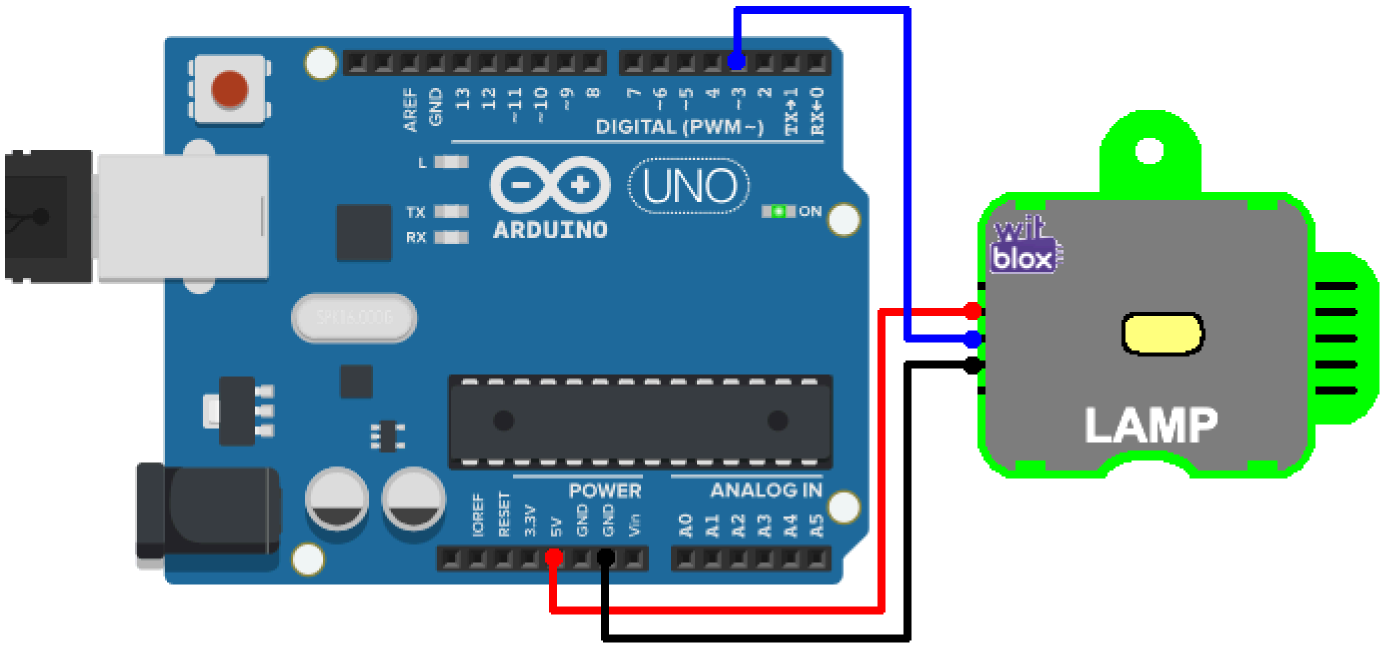
Arduino Code:

Now that we have already understood how dimming works for a single LED, it is very easy to imagine as well as understand how multiple LEDs will work with dimming. Here is the code for controlling 3 LEDs with dimming.

```
#define grn 6                  // defining pin 6 as "grn"
#define ylw 5                  // defining pin 5 as "ylw"
#define red 3                  // defining pin 3 as "red"

void setup()
{
  pinMode(grn, OUTPUT);        // declaring "grn" pin as output
  pinMode(ylw, OUTPUT);      // declaring "ylw" pin as output
  pinMode(red, OUTPUT);      // declaring "red" pin as output
}

void loop()
{
  for(int i=0;i<255;i++)
    {
      analogWrite(grn,i);
      delay(20);
    }
  analogWrite(grn,0);

  for(int j=0;j<255;j++)
    {
      analogWrite(ylw,j);
      delay(20);
    }
  analogWrite(ylw,0);

  for(int k=0;k<255;k++)
    {
      analogWrite(red,k);
      delay(20);
    }
  analogWrite(red,0);
}
```

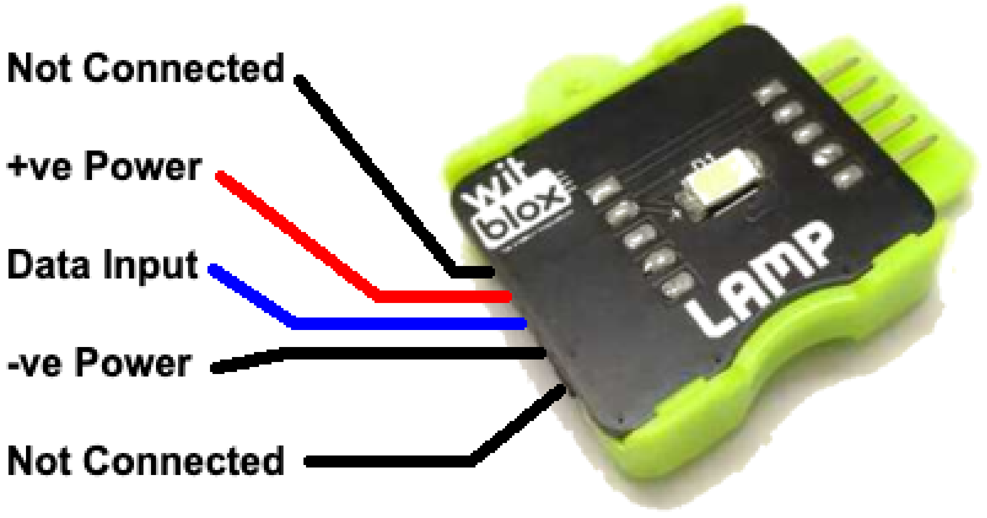
Part 3:

Connection Diagram:



Explanation:

In the third part of this activity, we take the Witblox Lamp blox and connect it to the Arduino board using the breadboard and some M - M wires as shown in the above connection diagram and control its brightness using an Arduino program.



Here are the connections:

1. The second pin from top on the Lamp blox is the 5v pin (+ve power pin) and is connected to the +ve line of the breadboard. This is the Red wire shown above, connecting the breadboard to Lamp blox.

2. The second pin from the bottom on the Lamp blox is the Gnd pin (-ve power pin) and is connected to the Gnd line of the breadboard. This is the Black wire shown above, connecting the breadboard to Lamp blox.

3. The centre pin on the Lamp blox is the Data input pin and is connected to the Arduino's digital pin 3 (D3). This is the Blue wire from Arduino to Lamp blox.

4. And as usual, the 5v pin on the Arduino is connected to the +ve line of the breadboard and the Gnd pin on the Arduino is connected to the -ve line of the breadboard.

Arduino Code:

The Arduino program for this third part is exactly the same program that we have used in Part1 as is clear from the code mentioned below. The only difference is that the name of the pin has changed to "Lamp" as we are connecting the Lamp blox.

The connections are somewhat different from the way normal LEDs are connected, as mentioned in the previous section. This is because the Lamp blox has its own internal LED driver circuit and therefore requires external power from the Arduino to operate.

Here is the Arduino code for controlling the Lamp blox with dimming.

```
#define Lamp 3                  // define pin 3 as "Lamp"

void setup()                    // execute just once
{
  pinMode(Lamp, OUTPUT);        // declare "Lamp" pin as output
}

void loop()                     // keep executing forever
{
  for(int i=0;i<255;i++)
    {
      analogWrite(Lamp,i);      // write Lamp with dimming level
      delay(20);                // wait for sometime
```

```
        }
}
```

Explanation:

Again, no different from the explanation of Part 1 or 2, we have the “setup” function for declaring the pin as output type, the “loop” function for repeating the dimming process and the “for” loop construct for dividing the brightness into 255 levels with each level held for a 20 ms duration.

The only difference from the previous code is that the name of the pin has changed from “red” to “Lamp” and that we are using a single output instead of three.

Outcome and Observations:

1. For Part 1, once you are done compiling and uploading the code to the Arduino Nano, you will observe the following:
    a. The Red LED gradually increases in brightness and reaches full brightness.
    b. Once the LED has attained full brightness, it goes off and starts over again.
2. Try increasing the numbers inside the delay function from 20 to 40. You will see that the total duration of the dimming has doubled. This is because every individual level now stays for 40 ms instead of 20 and thus the total time to reach full brightness has gone from

20 x 255 = 5100 ms   to   40 x 255 = 10200 ms.

3. For Part 2, once you are done compiling and uploading the code to the Arduino Nano, you will observe the following:
    a. The Green LED goes from zero to full brightness in just above 5 seconds and then turns Off.

   b. The Yellow LED goes from zero to full brightness in just above 5 seconds and then turns Off.
   c. The Red LED goes from zero to full brightness in just above 5 seconds and then turns Off.
   d. And then the entire process repeats itself.
4. Try replacing the entire “loop” function with the following code.

```
void loop()
    {
        for(int i=0;i<255;i++)
            {
                analogWrite(grn,i);
                analogWrite(ylw,i);
                analogWrite(red,i);
                delay(20);
            }
    }
```

5. For Part 3, once you are done compiling and uploading the code to the Arduino, you will observe the same behavior as Part 1. But what if you want the Lamp to go back to zero brightness in the same 255 steps. Figure this part out on your own. Happy brainstorming.