



Project 8

Mood Lamp

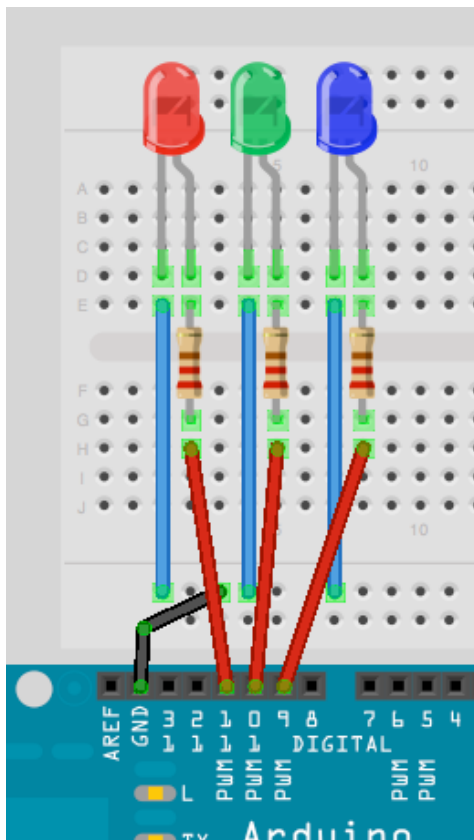
Project 8 – Mood Lamp

In the last project we saw that we could adjust the brightness of an LED using the PWM capabilities of the Atmega chip. We will now take advantage of this capability by using a red, green and blue LED and by mixing their colours to create any colour we wish. From that, we will create a mood lamp similar to the kind you see for sale all over the place nowadays.

What you will need

Red Clear LED	
Green Clear LED	
Blue Clear LED	
3 x 220Ω Resistor	

Connect it up



Get a piece of paper about A5 size, roll it into a cylinder then tape it so it remains that way. Then place the cylinder over the top of the 3 LED's.

Enter the code

```
// Project 8 – Mood Lamp
float RGB1[3];
float RGB2[3];
float INC[3];

int red, green, blue;

int RedPin = 11;
int GreenPin = 10;
int BluePin = 9;

void setup()
{
  Serial.begin(9600);
  randomSeed(analogRead(0));

  RGB1[0] = 0;
  RGB1[1] = 0;
  RGB1[2] = 0;

  RGB2[0] = random(256);
  RGB2[1] = random(256);
  RGB2[2] = random(256);
}

void loop()
{
  randomSeed(analogRead(0));

  for (int x=0; x<3; x++) {
    INC[x] = (RGB1[x] - RGB2[x]) / 256; }

  for (int x=0; x<256; x++) {

    red = int(RGB1[0]);
    green = int(RGB1[1]);
    blue = int(RGB1[2]);

    analogWrite (RedPin, red);
    analogWrite (GreenPin, green);
    analogWrite (BluePin, blue);
    delay(100);

    RGB1[0] -= INC[0];
    RGB1[1] -= INC[1];
    RGB1[2] -= INC[2];
  }
  for (int x=0; x<3; x++) {
    RGB2[x] = random(556)-300;
    RGB2[x] = constrain(RGB2[x], 0, 255);
    delay(1000);
  }
}
```

When you run this you will see the colours slowly change. You've just made your own mood lamp.

Project 8 – Code Overview

The LED's that make up the mood lamp are red, green and blue. In the same way that your computer monitor is made up of tiny red, green and blue (RGB) dots, the map can generate different colours by adjusting the brightness of each of the 3 LED's in such a way to give us a different RGB value.

An RGB value of 255, 0, 0 would give us pure red. A value of 0, 255, 0 would give pure green and 0, 0, 255 pure blue. By mixing these we can get any colour we like with This is the additive colour model. If you were just turn the LED's ON or OFF (i.e. Not have different brightnesses) you would still get different colours as in this table.

Red	Green	Blue	Colour
255	0	0	Red
0	255	0	Green
0	0	255	Blue
255	255	0	Yellow
0	255	255	Cyan
255	0	255	Magenta
255	255	255	White

By adjusting the brightnesses using PWM we can get every other colour in between too. By placing the LED's close together and by mixing their values, the light spectra of the 3 colours added together make a single colour. By diffusing the light with our paper cylinder we ensure the colours are mixed nicely. The LED's can be placed into any object that will diffuse the light or you can bounce the light off a reflective diffuser. Try putting the lights inside a ping pong ball or a small white plastic bottle (the thinner the plastic the better).

The total range of colours we can get using PWM with a range of 0 to 255 is 16,777, 216 colours (256x256x256) which is way more than we would ever need.

In the code, we start off by declaring some floating point arrays and also some integer variables that will store our RGB values as well as an increment value.

```
float RGB1[3];
float RGB2[3];
float INC[3];

int red, green, blue;
```

In the setup function we have

```
randomSeed(analogRead(0));
```

The randomSeed command is used for creating random (actually pseudo-random) numbers. Computer chips are not able to produce truly random numbers so they tend to look at data in a part of it's memory that may differ or look at a table of different values and use those as a pseudo-random number. By setting a 'seed', you can tell the computer where in memory or in that table to start counting from. In this case the value we give to the randomSeed is a value read from Analog Pin 0. As we don't have anything connected to Analog Pin 0 all we will read is a random number created by analog noise.

Once we have set a 'seed' for our random number we can create one using the random() function. We then have two sets of RGB values stored in a 3 element array. RGB1 is the RGB values we want the lamp to start with (in this case all zeros or off).

```
RGB1[0] = 0;
RGB1[1] = 0;
RGB1[2] = 0;
```

Then the RGB2 array is a set of random RGB values that we want the lamp to transition to,

```
RGB2[0] = random(256);
RGB2[1] = random(256);
RGB2[2] = random(256);
```

In this case we have set them to a random number set by random(256) which will give is a number between 0 and 255 inclusive (as the number will always range from zero upwards).

If you pass a single number to the random() function then it will return a value between 0 and 1 less than the number, e.g. random(1000) will return a number between 0 and 999. If you supply two numbers as it's parameters then it will return a random number between the lower number inclusive and the maximum number (-1). E.g. random(10,100) will return a random number between 10 and 99.

In the main program loop we first take a look at the start and end RGB values and work out what value is needed as an increment to progress from one value to the other in 256 steps (as the PWM value can only be between 0 and 255). We do this with

```
for (int x=0; x<3; x++) {
    INC[x] = (RGB1[x] - RGB2[x]) / 256; }
```

This for loop sets the INCrement values for the R, G and B channels by working out the difference between the two brightness values and dividing that by 256.

We then have another for loop

```
for (int x=0; x<256; x++) {

    red = int(RGB1[0]);
    green = int(RGB1[1]);
    blue = int(RGB1[2]);

    analogWrite (RedPin, red);
    analogWrite (GreenPin, green);
    analogWrite (BluePin, blue);
    delay(100);

    RGB1[0] -= INC[0];
    RGB1[1] -= INC[1];
    RGB1[2] -= INC[2];
}
```

and this sets the red, green and blue values to the values in the RGB1 array, writes those values to pins 9, 10 and 11, then deducts the increment value then repeats this process 256 times to slowly fade from one random colour to the next. The delay of 100ms in between each step ensures a slow and steady progression. You can of course adjust this value if you want it slower or faster or you can add a potentiometer to allow the user to set the speed.

After we have taken 256 slow steps from one random colour to the next, the RGB1 array will have the same values (nearly) as the RGB2 array. We now need to decide upon another set of 3 random values ready for the next time. We do this with another for loop

```
for (int x=0; x<3; x++) {
    RGB2[x] = random(556)-300;
    RGB2[x] = constrain(RGB2[x], 0, 255);
    delay(1000);
}
```

The random number is chosen by picking a random number between 0 and 556 (256+300) and then deducting 300. The reason we do that is to try and force primary colours from time to time to ensure we don't always just get pastel shades. We have 300 chances out of 556 in getting a negative number and therefore forcing a bias towards one or more of the other two colour channels. The next command makes sure that the numbers sent to the PWM pins are not negative by using the constrain() function.

The constrain function requires 3 parameters - x, a and b as in constrain(x, a, b) where x is the number we want to constrain, a is the lower end of the range and b is the higher end. So, the constrain functions looks at the value of x and makes sure it is within the range of a to b. If it is lower than a then it sets it to a, if it is higher than b it sets it to b. In our case we make sure that the number is between 0 and 255 which is the range of our PWM output.

As we use random(556)-300 for our RGB values, some of those values will be lower than zero and the constrain function makes sure that the value sent to the PWM is not lower than zero.

Forcing a bias towards one or more of the other two channels ensures more vibrant and less pastel shades of colour and also ensures that from time to time one or more channels are turned off completely giving a more interesting change of lights (or moods).

Exercise

See if you can make the lights cycle through the colours of the rainbow rather than between random colours.