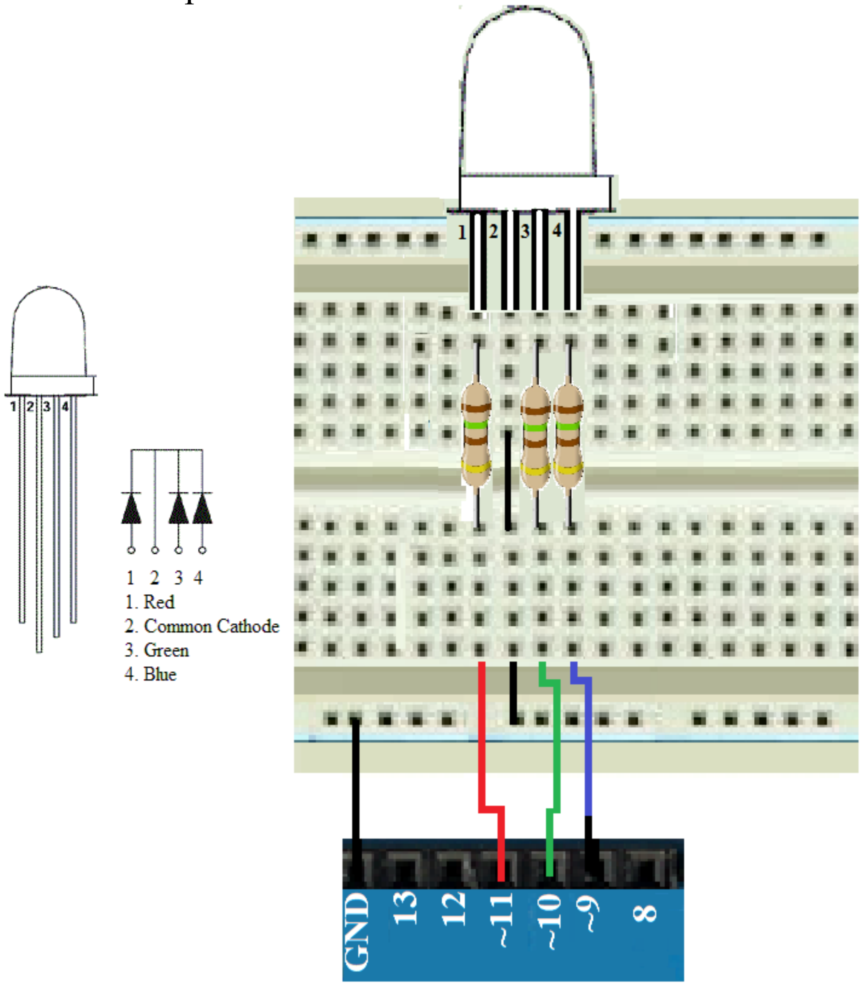
Controlling The Real World With Computers

::. Control And Embedded Systems .::

Putting It All Together - Controlling The Hardware With The Software
Programming Experiment 1

We will next take a look at a way to make an interesting display using a RGB (red, green, blue) LED which is actually three LEDs in a single package. It contains a red, green and blue LED. Almost 17,000,000 colors can be obtained from it by varying the brightness of each of the primary colors it contains. The one used in this experiment, along with how to hook it up looks like this:

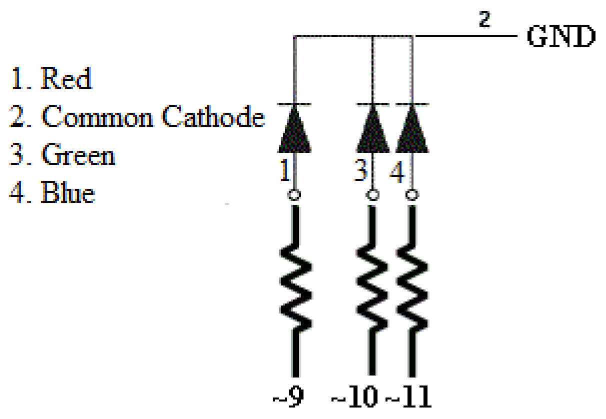


Notice that the cathodes of the three LEDs are common in the package, which. should be connected to ground on the UNO.   You can get the RGB LEDs from Jameco Electronics here

Connect # 3 on the LED through a 150 ohm resistor to 9~ on the UNO,
# 4 through a 150 ohm resistor to 10~ on the UNO and # 1 through a 150 ohm resistor to 11~ on the UNO Now load BrightTest here, compile and run it. It looks like this:

```
/*
BrightTest
Turns one color on 1 second, then next, etc.
*/

int Red_Pin_9 = 9;
int Green_Pin_10 = 10;
int Blue_Pin_11 = 11;

void setup()
{
	pinMode(Red_Pin_9, OUTPUT);
	pinMode(Green_Pin_10, OUTPUT);
	pinMode(Blue_Pin_11, OUTPUT);
}

int RedOn = 255, GreenOn = 255, BlueOn = 255;
int RedOff = 0, GreenOff = 0, BlueOff = 0;

void loop()
{
	SetLED(RedOn, GreenOff, BlueOff, 1000); // red on, green off, blue off 1 second
	SetLED(RedOff, GreenOn, BlueOff, 1000); // red off, green on, blue off 1 second
	SetLED(RedOff, GreenOff, BlueOn, 1000); // red off, green off, blue on 1 second
	SetLED(RedOn, GreenOn, BlueOn, 3000); // all on 3 seconds

} // end void loop()

void SetLED(int RedValue, int GreenValue, int BlueValue, int DelayTime)
{
	analogWrite(Red_Pin_9, RedValue);
	analogWrite(Green_Pin_10, GreenValue);
	analogWrite(Blue_Pin_11, BlueValue);
	delay(DelayTime);

} // end void SetLED(..)
```

This test turns each LED on for 1 second, then all three on for 3 seconds. Notice the brightness of each color. Note that, like in the blink program, the pin is made high to turn an LED on. Since these are PWM analog pins, 0 is full low and 255 is full high. Everything between the two provides a brightness in proportion to the value:



Making the current limiting resistor of a color high turns that color on. You might note that red is not quite as bright as green and blue. Increase its brightness by decreasing its resistor from 150 to 100 ohms. The text for ThreeSweep begins below. The comments will hopefully show how the various variables and sections work.

```
/*
  ThreeSweep (it's a good idea to put the name of your program at the top and bottom)
 Gradually fades on 3 LEDs on then fades them off in sequence
 Copyright 2013 Joe D. Reeder
 */

int Red_Pin = 11;
int Green_Pin = 10;
int Blue_Pin = 9;

void setup()
{
  pinMode(Red_Pin, OUTPUT);
  pinMode(Green_Pin, OUTPUT);
  pinMode(Blue_Pin, OUTPUT);
}
```

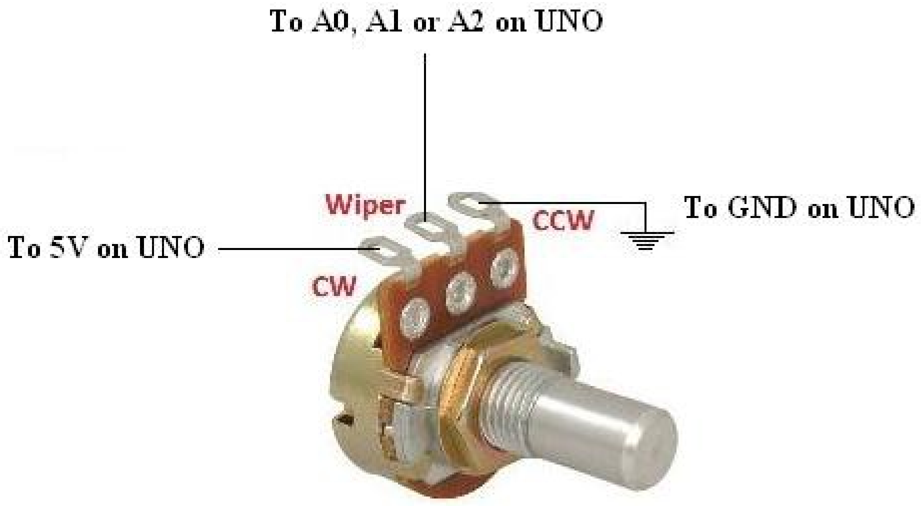
```
/*
experiment with the following value integers by setting them to different values
an integer (int) is simply a variable that can be set to various positive and negative values
There are a few rules for variables and other labels in a C program:

1. Every variable name in C must start with a letter.
2. The rest of the name can consist of letters, numbers and underscore characters.
3. The use of a blank space in a label name is not permitted.  For example,  The label
   this is a name is not allowed, but this_is_a_name and ThisIsAName are OK.
   Don't worry, the compiler will let you know if it doesn't like a name.
   Note that setup() and loop() know nothing about variables declared after setup() and loop().
   The variables are said to be out of scope for setup() and/or loop().
*/

// experiment with these value numbers to see what happens
int RedValue = 0; // start red full off
int GreenValue = 200; // start green almost full on (full on is 255)
int BlueValue = 127; // start blue half on

// The step values determine how fast a color value will change (larger makes faster)
// Positive numbers cause the color values to increase
// Negative numbers cause the color values to decrease
// Experiment with the following step values to see what happens
int RedStep = 3;
int GreenStep = -5;
int BlueStep = 4;

//  changing the delay can cause some interesting effects – experiment
int DelayTime = 20;
```

```
/*
   The following method allows dynamic control of changing the step
   direction and magnitude.  The variations help provide the interesting
   overlap of colors.  The result is not simple R,G,B but the mixture of
   the three to provide many more colors.  The number of possible colors
   is huge: 256*256*256 = 16,777,216.
*/
void loop()
{

          //  analogWrite will cause the  LED Pin to output a pulse wave.
          //  The larger the number, the longer the ON part of the wave.
          //  If, for example, the pulse is on 25% of the time, the LED will be at one fourth its
          //  full brightness. If it's on half the time, the LED will be at half its full brightness.
          //  A value of 255 is full on and a value of 0 is full off.
          analogWrite(Red_Pin, RedValue); // write the values to the pins
          analogWrite(Green_Pin, GreenValue);
          analogWrite(Blue_Pin, BlueValue);

         // The += below means add the value of right-hand variable to the variable on the left.
         // It adds the step value to the color value.  The color value is increased when the step
         // value is positive and decreased when it is negative.
          RedValue += RedStep;
          GreenValue += GreenStep;
          BlueValue += BlueStep;

          // The *= -1 below means to multiply the step value by -1.
          //  It changes a positive value to a negative, and a negative value to a positive.

          // red
          if (RedValue > 255)             // over 255 so set to 254 & reverse step direction with *= -1
          {
                    RedValue = 254;
                    RedStep *= -1;
          }

          else if(RedValue < 0)           // less than 0 so set to 1 & reverse step direction with *= -1
          {
                    RedValue = 1;
                    RedStep *= -1;
          }
```

```
        // green
        if(GreenValue > 255)          // over 255 so set to 254 & reverse step direction with *= -1
        {
                GreenValue = 254;
                GreenStep *= -1;
        }

        else if(GreenValue < 0)       // less than 0 so set to 1 & reverse step direction with *= -1
        {
                GreenValue = 1;
                GreenStep *= -1;
        }


        // blue
        if(BlueValue > 255)           // over 255 so set to 254 & reverse step direction with *= -1
        {
                BlueValue = 254;
                BlueStep *= -1;
        }

        else if(BlueValue < 0)        // less than 0 so set to 1 & reverse step direction with *= -1
        {
                BlueValue = 1;
                BlueStep *= -1;
        }

  delay(DelayTime);

} // end void loop()

// end ThreeSweep
// (it's a good idea to put the name of your program at the top and bottom,
// especially at the end in case an editor drops part of your code)
```

Download ThreeSweep here

At the end of all of the tests, there is a delay by the number of milliseconds indicated by the DelayTime variable. Next, we will make a color changer. Hook up three potentiometers (such as from Jameco here) this way:



Load ColorChanger here and you will see something like the following:

```
/*
ColorChanger
Read analog values from potentiometers and output to color channels
*/

int Red_PWM_Pin_9 = 9;
int Green_PWM_Pin_10 = 10;
int Blue_PWM_Pin_11 = 11;

void setup()
{
	pinMode(Red_PWM_Pin_9, OUTPUT);
	pinMode(Green_PWM_Pin_10, OUTPUT);
	pinMode(Blue_PWM_Pin_11, OUTPUT);
}

int RedAnalog_Pin_A0 = A0; // read red analog from AD converter A0
int GreenAnalog_Pin_A1 = A1; // read green analog from AD converter A1
int BlueAnalog_Pin_A2 = A2; // read blue analog from AD converter A2

int RedAnalogValue = 0; // holds analog from AD A0
int GreenAnalogValue = 0; // holds analog from AD A1
int BlueAnalogValue = 0; // holds analog from AD A2
```

```
void loop()
{
        // Read the analog value and map to the analog variable
        // The analogRead() output will range from 0 to 1023
        // Map that to 0 to 255 for PWM using the built-in map function
        RedAnalogValue = map(analogRead(RedAnalog_Pin_A0), 0, 1023, 0, 255);
        GreenAnalogValue = map(analogRead(GreenAnalog_Pin_A1), 0, 1023, 0, 255);
        BlueAnalogValue = map(analogRead(BlueAnalog_Pin_A2), 0, 1023, 0, 255);

        analogWrite(Red_PWM_Pin_9, RedAnalogValue);
        analogWrite(Green_PWM_Pin_10, GreenAnalogValue);
        analogWrite(Blue_PWM_Pin_11, BlueAnalogValue);

} // end void loop()

// end ColorChanger
```

This is basically a simplified version of ThreeSweep. It accepts analog input for the three colors using analogRead(..) to read analog inputs A0, A1 and A2. This reads analog values from the potentiometers. The result will be a number from 0 to 1023, representing 0 to 5 volts at the analog input. The program uses the map function to map the 0 to 1023 value to the 0 to 255 values the PWM analog channels can use. The resulting value is sent to the respective PWM output

Use a combination of alligator clip leads and breadboard jumper wires to connect all clockwise terminals to +5 volts and all counterclockwise terminals to ground. Connect the wipers to A0 for the red control, A1 for green and A2 for blue. Compile ColorChanger. You should be able to change the intensity of each color by turning the shafts. A color should be off at with the control fully counterclockwise, and full on with it fully clockwise.

A challenge for next time:

Use the potentiometers to modify the step sizes for each color in ThreeSweep. Remember to keep the polarities and map to something like 1 to 10. Experiment (play).

Table Of Contents

Previous: Programming Part 3   Next: Programming Experiment 2