INDD 593 - Lab 1: RGB Night-Light

The Arduino IDE

In this lab, we will create an RGB Night-Light to gain exposure to the RGB LED in your Sparkfun Tinker kits. The RGB LED in the tinker kits can be used to create many multiple different colors that can be beneficial to projects you may come up with!

First we will get started with installing the Arduino IDE on your computer. Here is the link to the download below:

https://www.arduino.cc/en/software

This will be the main console where you can write code to your Sparkfun. All the code will be provided through the labs, so you may easily copy and paste everything into the IDE.

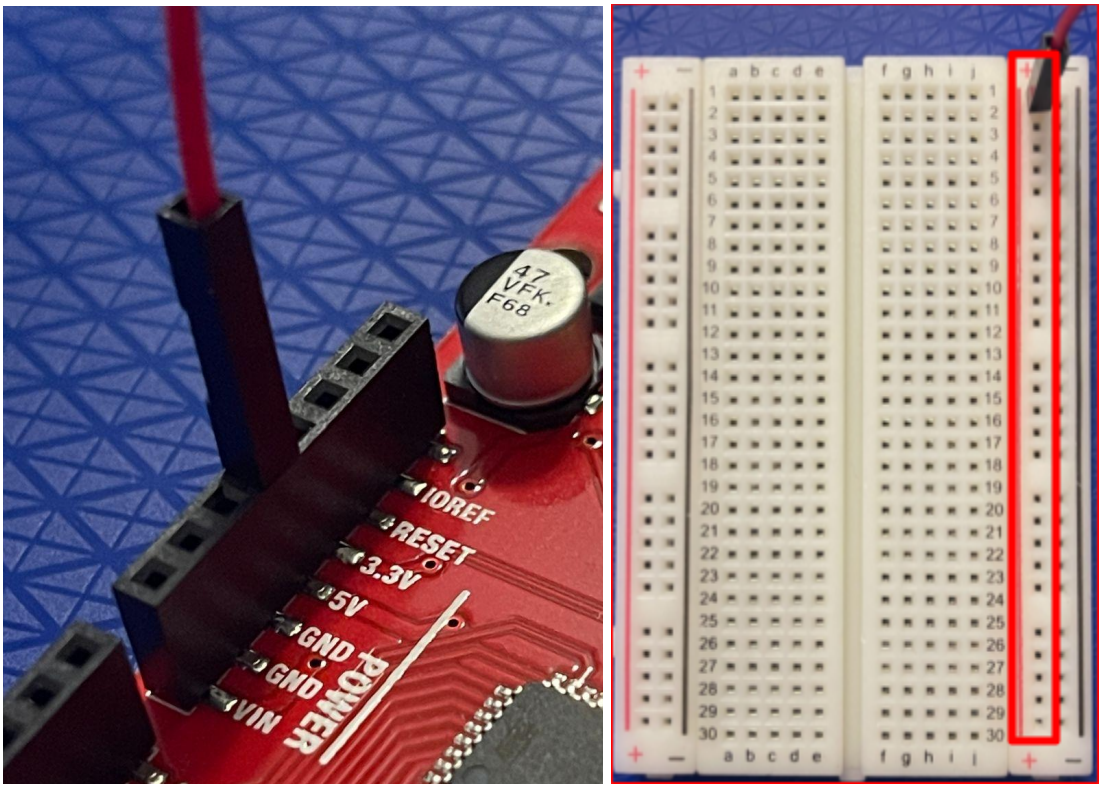
We will also have to download the CH340 Drivers to make the Sparkfun work with your computer. Please follow the guide in the link below:

https://learn.sparkfun.com/tutorials/how-to-install-ch340-drivers

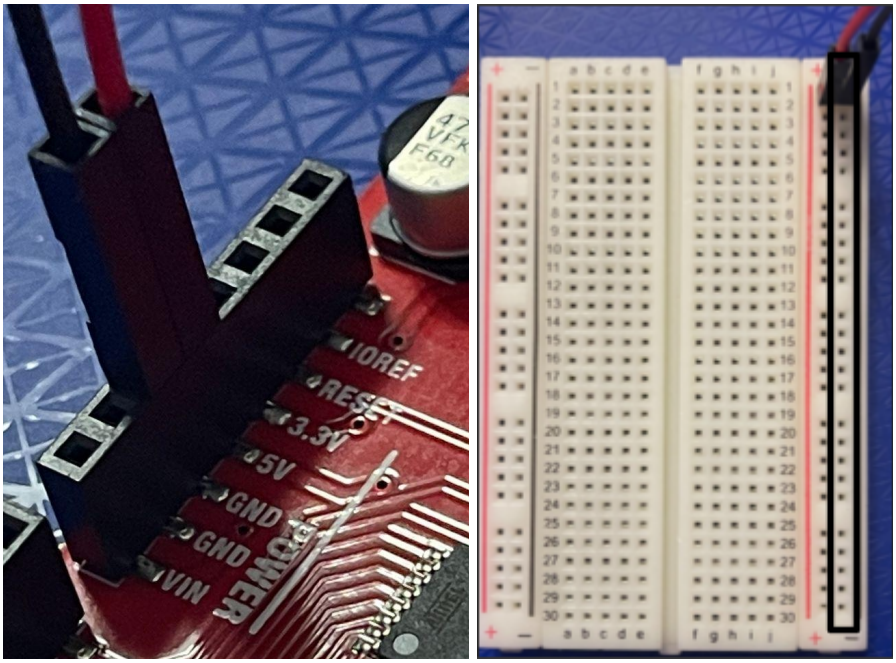
After you get the IDE installed we can continue on with more explanation of the breadboard within the tinker kit.

Breadboard

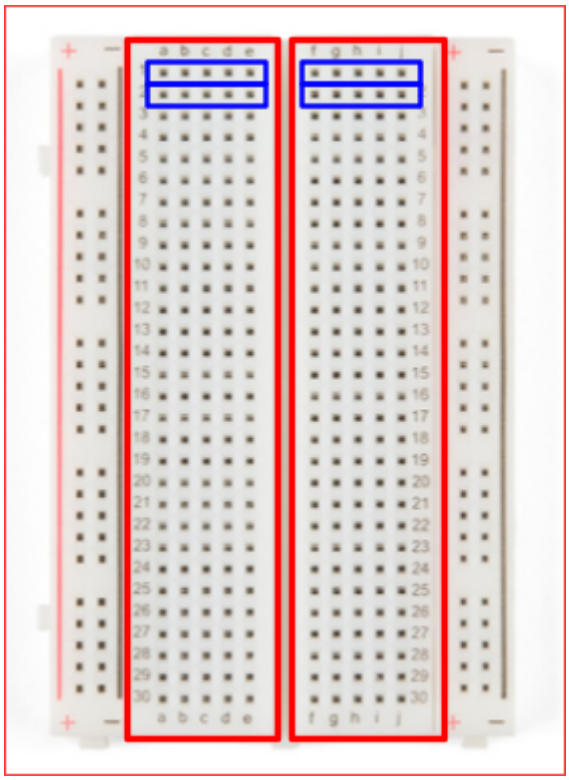
The breadboard is a communicator between the components and the Arduino. First thing we will go into is powering the breadboard with the Arduino. Almost all components need to be powered by electricity, to do this we have to connect a wire from the +5 V port (left side of the Arduino) to one of the +V ports on the breadboard.



After plugging in the +5V to a +V port on the breadboard, every port above within the red box is flowing with positive electricity. When a component receives positive electricity, it needs to go toward a negative (-) port on the board. This is where the GND port on your Arduino comes in. We can now attach this cable.



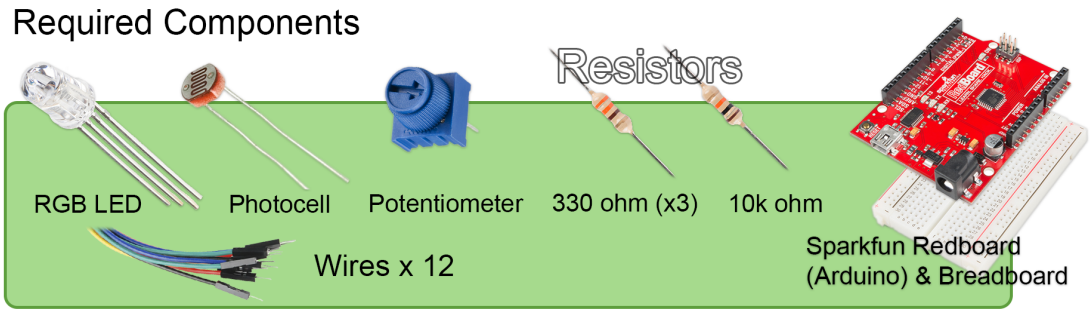
Just like the positive electricity, the negative ports on the board (box in black) are now grounded. Again, every component needs a flow from positive electricity to ground **(+ → -)**. Now you may be wondering, how can we attach a component to a wire? We can do this by all the open ports in the middle sections of the breadboard.



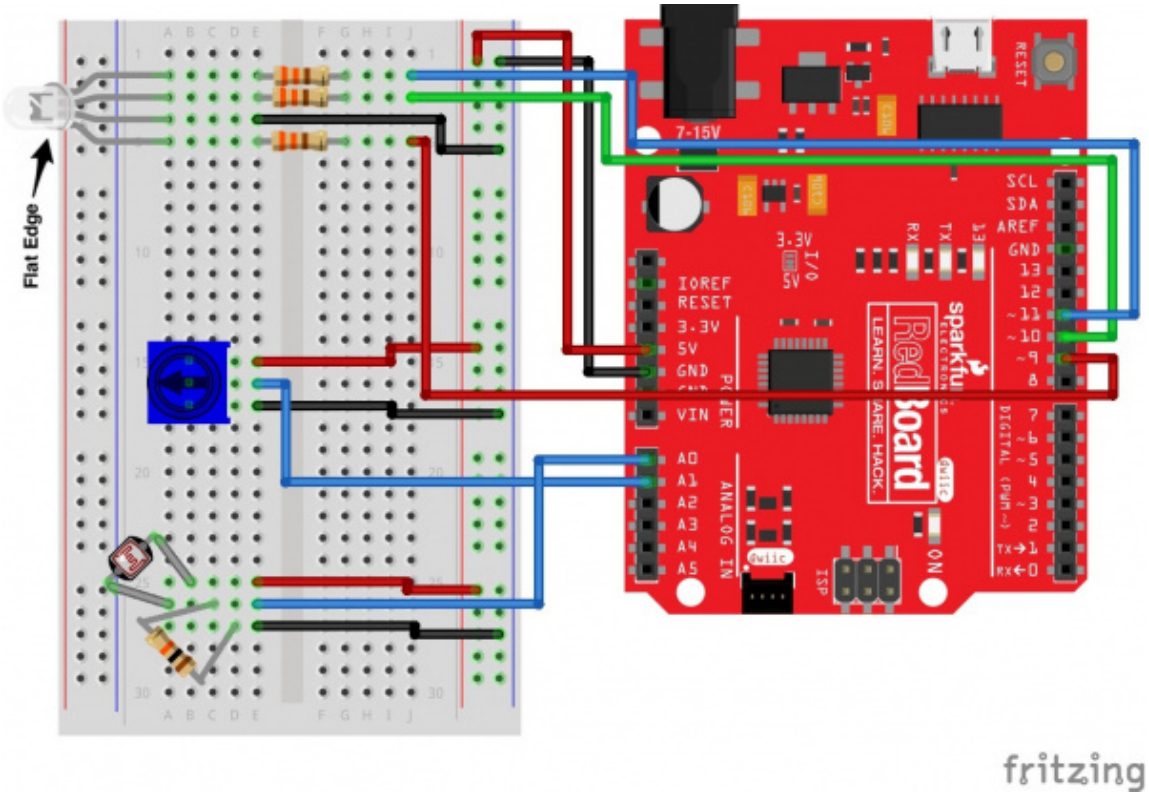
Sections a-e and f-j in the red boxes are completely separate from each other. The connecting components and wires within the blue boxes from a-e or f-j down the whole breadboard will connect them together. This will make more sense as we continue with the lab.

The RGB Night-Light

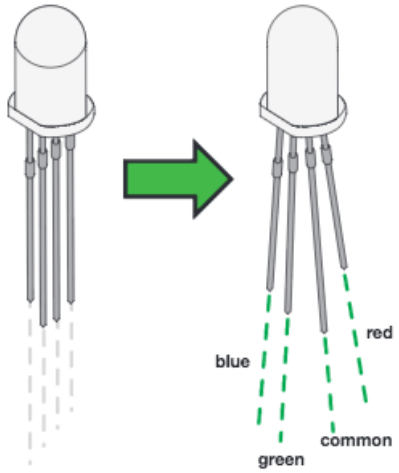
Now it's time for the fun part of the lab! We will use three different components within the tinker kit. This includes the following:



The RGB LED can change multiple different colors. The potentiometer is a dial that we will use to change to different colors. Finally, we have the photocell that will turn on or off the light depending if it is bright or dark in the room. The following is the schematic of the entire Night-light put together:

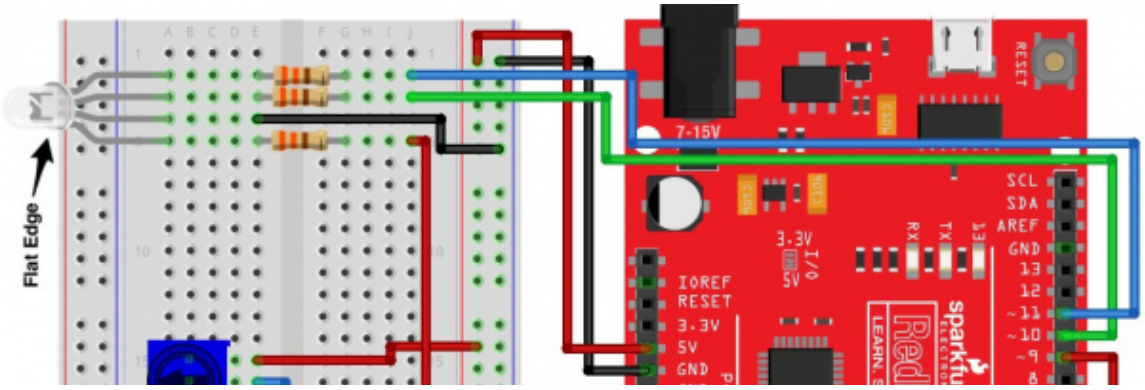


First, we will start building the schematic by attaching the RGB LED light.

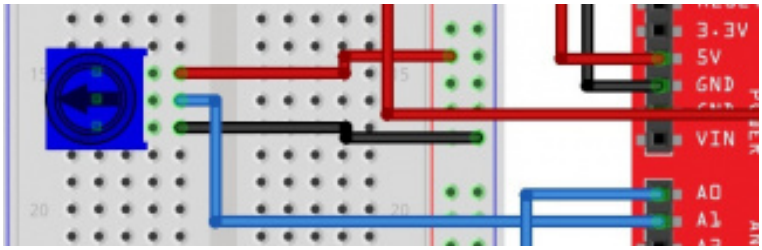


The blue, green, red, and common pins all vary in length. This is an easy way for us to tell which pin each color corresponds to. The blue, red, and green pins can be connected to any of the ports labeled 0-13 on the Arduino board. These 0-13 pins are giving the light positive electricity which is why we won't need to use the +V ports on the breadboard. The common pin is what we will be connecting to ground (since every component that has positive electricity must be connected to a negative (GND)).

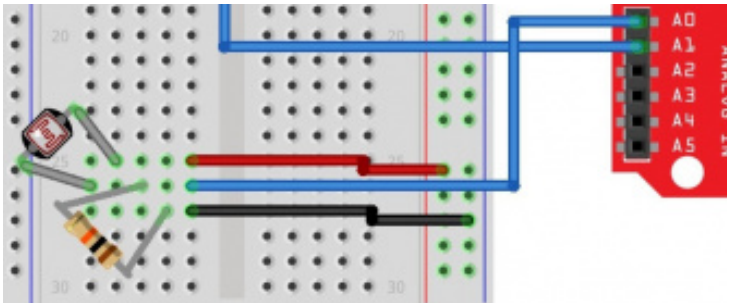
You are provided code (which already sets up the 0-13 ports) so we will use the above schematic that already has the pins set up where they need to be. Please connect the +5V, GND, and LED with the following picture below to help guide you through (blue should be the top pin and red should be the bottom pin):



Now we are going to plug in the potentiometer (dial) to our breadboard. Again, this is used to change the colors of the night light based on where the dial is turned.

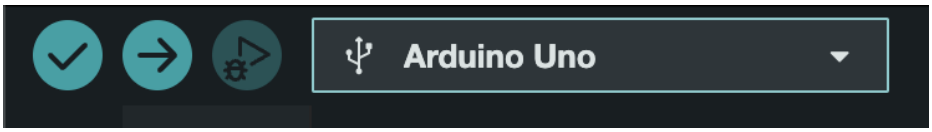


Finally, we will connect the photocell to our breadboard. Again, this is used to sense whether it is light or dark within the room.

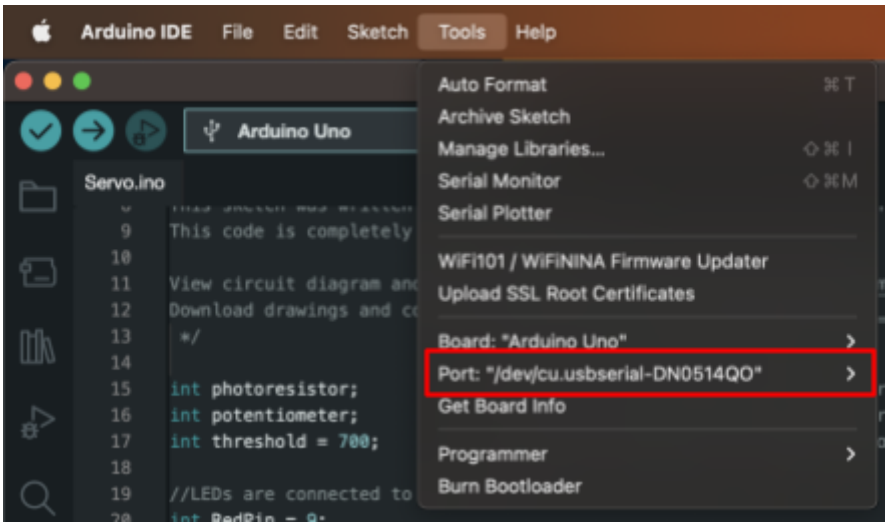


Now we can power on the Arduino with our provided red power cable:

The Mini-USB will connect to the Arduino and the USB should be connected to your PC/Laptop. Now we can open the Arduino IDE. We need to make sure the IDE is setup correctly to communicate with our Arduino. First, make sure that the board is setup as an Arduino Uno.



Now up on the task bar, there should be a tool option. Click that. Now go down to the Port option and make sure the port is talking to the Arduino. For me, it looks like this:



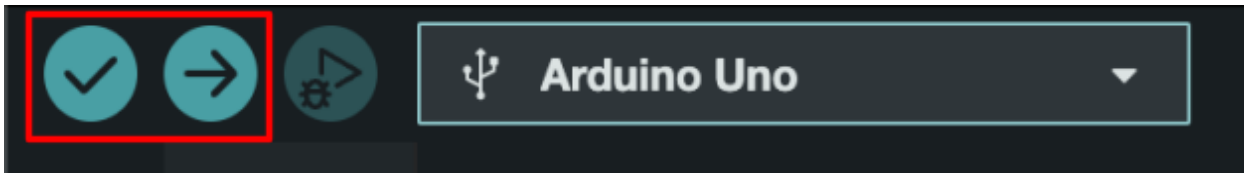
Now we can copy and paste the following code into the IDE:
The following is the code we will be using provided by Sparkfun.

```
/*
SparkFun Tinker Kit
Circuit 4: RGB Night-Light

Turns an RGB LED on or off based on the light level read by a photoresistor.
Change colors by turning the potentiometer.

This sketch was written by SparkFun Electronics, with lots of help from the Arduino community.
This code is completely free for any use.

View circuit diagram and instructions at: https://learn.sparkfun.com/tutorials/activity-guide-for-sparkfun-tinker-kit/
Download drawings and code at: https://github.com/sparkfun/SparkFun_Tinker_Kit_Code
 */

int photoresistor;         //variable for storing the photoresistor value
int potentiometer;         //variable for storing the photoresistor value
int threshold = 700;          //if the photoresistor reading is lower than this value the light wil turn on

//LEDs are connected to these pins
int RedPin = 9;
```

```
int GreenPin = 10;
int BluePin = 11;

void setup() {
  Serial.begin(9600);          //start a serial connection with the computer

  //set the LED pins to output
  pinMode(RedPin,OUTPUT);
  pinMode(GreenPin,OUTPUT);
  pinMode(BluePin,OUTPUT);
}

void loop() {

  photoresistor = analogRead(A0);         //read the value of the photoresistor
  potentiometer = analogRead(A1);

  Serial.print("Photoresistor value:");
  Serial.print(photoresistor);          //print the photoresistor value to the serial monitor
  Serial.print("  Potentiometer value:");
  Serial.println(potentiometer);          //print the photoresistor value to the serial monitor

  if(photoresistor < threshold){          //if it's dark (the photoresistor value is below the threshold) turn the LED on
    //These nested if staments check for a variety of ranges and
    //call different functions based on the current potentiometer value.
    //Those functions are found at the bottom of the sketch.
    if(potentiometer > 0 && potentiometer <= 150)
      red();
    if(potentiometer > 150 && potentiometer <= 300)
      orange();
    if(potentiometer > 300 && potentiometer <= 450)
      yellow();
    if(potentiometer > 450 && potentiometer <= 600)
      green();
    if(potentiometer > 600 && potentiometer <= 750)
      cyan();
    if(potentiometer > 750 && potentiometer <= 900)
      blue();
    if(potentiometer > 900)
      magenta();
  }
  else {                          //if it isn't dark turn the LED off

    turnOff();                        //call the turn off function

  }

  delay(100);                          //short delay so that the printout is easier to read
}

void red (){

    //set the LED pins to values that make red
    analogWrite(RedPin, 100);
```

```
  analogWrite(GreenPin, 0);
  analogWrite(BluePin, 0);
}
void orange (){

  //set the LED pins to values that make orange
  analogWrite(RedPin, 100);
  analogWrite(GreenPin, 50);
  analogWrite(BluePin, 0);
}
void yellow (){

  //set the LED pins to values that make yellow
  analogWrite(RedPin, 100);
  analogWrite(GreenPin, 100);
  analogWrite(BluePin, 0);
}
void green (){

  //set the LED pins to values that make green
  analogWrite(RedPin, 0);
  analogWrite(GreenPin, 100);
  analogWrite(BluePin, 0);
}
void cyan (){

  //set the LED pins to values that make cyan
  analogWrite(RedPin, 0);
  analogWrite(GreenPin, 100);
  analogWrite(BluePin, 100);
}
void blue (){

  //set the LED pins to values that make blue
  analogWrite(RedPin, 0);
  analogWrite(GreenPin, 0);
  analogWrite(BluePin, 100);
}
void magenta (){

  //set the LED pins to values that make magenta
  analogWrite(RedPin, 100);
  analogWrite(GreenPin, 0);
  analogWrite(BluePin, 100);
}
void turnOff (){

  //set all three LED pins to 0 or OFF
  analogWrite(RedPin, 0);
  analogWrite(GreenPin, 0);
  analogWrite(BluePin, 0);
}
```

After this is in pasted in the IDE, we can test to see if it works. Make sure to click the Verify button, then click the upload button:



Did the LED light up? If it didn't, try covering the photocell with your finger. Try turning the potentiometer or the dial. Can you see that it is changing colors? Pretty cool! Nice job completing the lab!

This is the first lab so please if you have any questions or feedback please let us know! You can email us at sdmay23-48@iastate.edu.

(Optional) Understanding of how the code works with the Arduino

The following is going over the given code, so if you may see this LED useful in other projects you maybe able to implement it on your own!

```
int photoresistor;        //variable for storing the photoresistor value
int potentiometer;        //variable for storing the photoresistor value
int threshold = 700;        //if the photoresistor reading is lower than this value the light wil turn on
```

The following is initiallizing the photoresistor, potentiometer, and threshold values of the photocell. The lab was to create a night light which may not have worked as it should. When it is bright out, the light should turn off. When it is dark, the light should turn on. In my case of testing this lab, my light was on when I was in a bright room. If this is the case for you as well, try turning the threshold value of 700 down to a smaller number. This could be a matter of trial and error until it works the way we want it to. After changing the threshold number, we can reupload the code to the Arduino updating it by pressing the upload button.

```
//LEDs are connected to these pins
int RedPin = 9;
int GreenPin = 10;
int BluePin = 11;
```

Next we have the pin initialization. You can see that we have the red, green, and blue pins connected to the Arduino on ports 9, 10, and 11. For example, if we were to change the RedPin=9; value to Redpin=0;. The RGB light would no longer have the red input going to the light. We can fix this by moving our wire on our Arduino from the 9 pin to the 0 pin. Then the RGB LED should work as expected.

```
void loop() {

  photoresistor = analogRead(A0);        //read the value of the photoresistor
```

```
potentiometer = analogRead(A1);

Serial.print("Photoresistor value:");
Serial.print(photoresistor);          //print the photoresistor value to the serial monitor
Serial.print("  Potentiometer value:");
Serial.println(potentiometer);          //print the photoresistor value to the serial monitor
```

With any start to an Arduino project, we will need to start with a void loop(){ (code) }. This just makes sure that the Arduino is constantly looping through each line of code making sure if anything changes, it will be updated in real time. Take for example, changing the potentiometer (dial) value. If this wasn't contained in the void loop, the Arduino would keep it at the starting constant color. With the potentiometer code set within the void loop, it will keep checking to see if the potentiometer's value has been changed.

The photoresistor and potentiometer analogRead(A#) values are initialized here. We have the photoresistor and potentiometer connected to the A0 and A1 ports on the Arduino. The reason they are connected to these ports is because they are constantly changing values which need to be set by Analog ports.

The rest of the code is the Serial.print, this isn't really necessary to our lab, but it just lets the user know what values the photoresistor and potentiometer are reading on the Arduino IDE.

```
if(photoresistor < threshold){
  if(potentiometer > 0 && potentiometer <= 150)
    red();
  if(potentiometer > 150 && potentiometer <= 300)
    orange();
  if(potentiometer > 300 && potentiometer <= 450)
    yellow();
  if(potentiometer > 450 && potentiometer <= 600)
    green();
  if(potentiometer > 600 && potentiometer <= 750)
    cyan();
  if(potentiometer > 750 && potentiometer <= 900)
    blue();
  if(potentiometer > 900)
    magenta();
}
```

Next we have the if statement. The following if(photoresistor < threshold) is stating that if the photoresistor is reading a brightness value of less than the threshold value you stated above, the light will turn on with the following code:

```
  if(potentiometer > 0 && potentiometer <= 150)
    red();
  if(potentiometer > 150 && potentiometer <= 300)
    orange();
  if(potentiometer > 300 && potentiometer <= 450)
    yellow();
```

```
  if(potentiometer > 450 && potentiometer <= 600)
    green();
  if(potentiometer > 600 && potentiometer <= 750)
    cyan();
  if(potentiometer > 750 && potentiometer <= 900)
    blue();
  if(potentiometer > 900)
    magenta();
```

This code is checking the value of the potentiometer. The potentiometer contains values 0-1023. So for example if the potentiometer is turned to a value between 0-150 the color will change to red:

```
if(potentiometer > 0 && potentiometer <= 150)
    red();
```

If the potentiometer is turned to a value between 151-300 the color will change to orange:

```
if(potentiometer > 150 && potentiometer <= 300)
    orange();
```

And so on.

```
 else {
  turnOff();
 }
```

Next we have the else statement. The else statement only works after an if statement. We have to now understand that if the following statement is not true if(photoresistor < threshold){ } it will now go to the else statement which is turnOff();. Like what was said above, if the photoresistor is greater than the threshold then the LED will turn off. Meaning that the photocell senses that you are in a bright room.

```
delay(100);
```

The delay just adds a little more time for the value of the potentiometer and photocell to print out their values for the user to read within the IDE.

```
void red (){

  //set the LED pins to values that make red
  analogWrite(RedPin, 100);
  analogWrite(GreenPin, 0);
  analogWrite(BluePin, 0);
}
void orange (){

  //set the LED pins to values that make orange
  analogWrite(RedPin, 100);
  analogWrite(GreenPin, 50);
```

```
  analogWrite(BluePin, 0);
}
void yellow (){

  //set the LED pins to values that make yellow
  analogWrite(RedPin, 100);
  analogWrite(GreenPin, 100);
  analogWrite(BluePin, 0);
}
void green (){

  //set the LED pins to values that make green
  analogWrite(RedPin, 0);
  analogWrite(GreenPin, 100);
  analogWrite(BluePin, 0);
}
void cyan (){

  //set the LED pins to values that make cyan
  analogWrite(RedPin, 0);
  analogWrite(GreenPin, 100);
  analogWrite(BluePin, 100);
}
void blue (){

  //set the LED pins to values that make blue
  analogWrite(RedPin, 0);
  analogWrite(GreenPin, 0);
  analogWrite(BluePin, 100);
}
void magenta (){

  //set the LED pins to values that make magenta
  analogWrite(RedPin, 100);
  analogWrite(GreenPin, 0);
  analogWrite(BluePin, 100);
}
void turnOff (){

  //set all three LED pins to 0 or OFF
  analogWrite(RedPin, 0);
  analogWrite(GreenPin, 0);
  analogWrite(BluePin, 0);
}
```

Finally, we have the following color statements. These are the functions that are called when a certain potentiometer value is reached. Look back at the potentiometer if statements. Do you see when each of these colors are called?

Lets look at cyan for example.

```
void cyan (){

  //set the LED pins to values that make cyan
```

```
    analogWrite(RedPin, 0);
    analogWrite(GreenPin, 100);
    analogWrite(BluePin, 100);
}
```

This is just like what you may have learned in art class. Red and Yellow mixed together make orange. Well that may be true when painting, but RGB's work on a different scale than what you maybe used to. It maybe best to look up when you need to create a certain color using RGB on google to give you a confirmed answer on what values you may need to add to red, green, or blue. Cyan for example takes a combination of Green and Blue, with no value for Red. This can be seen above. Green and Blue have been set to values of 100, while Red has been set to 0. This is pretty interesting! You may have some LED lights in your house, this is a good example on how the color mixing works to give you a bunch of different colors!

```
void turnOff (){

    //set all three LED pins to 0 or OFF
    analogWrite(RedPin, 0);
    analogWrite(GreenPin, 0);
    analogWrite(BluePin, 0);
}
```

Finally, you have the turnOff function which sets all values to 0 turning off the LED. This is used above when the photocell value is greater than the threshold value (meaning you are in a bright room).

Thanks again for looking through this lab! If you made it this far and are still confused on anything shown above or have any feedback for us, let us know! Please email us at sdmay23-48@iastate.edu.

Extra notes on color:
Looking at the function corresponding to each color, you may notice that there is duplicated code that look the same (analogWrite RedPin, GreenPin, BluePin), only differing in the colors values set in each function (redPin 100 for red(), for example).