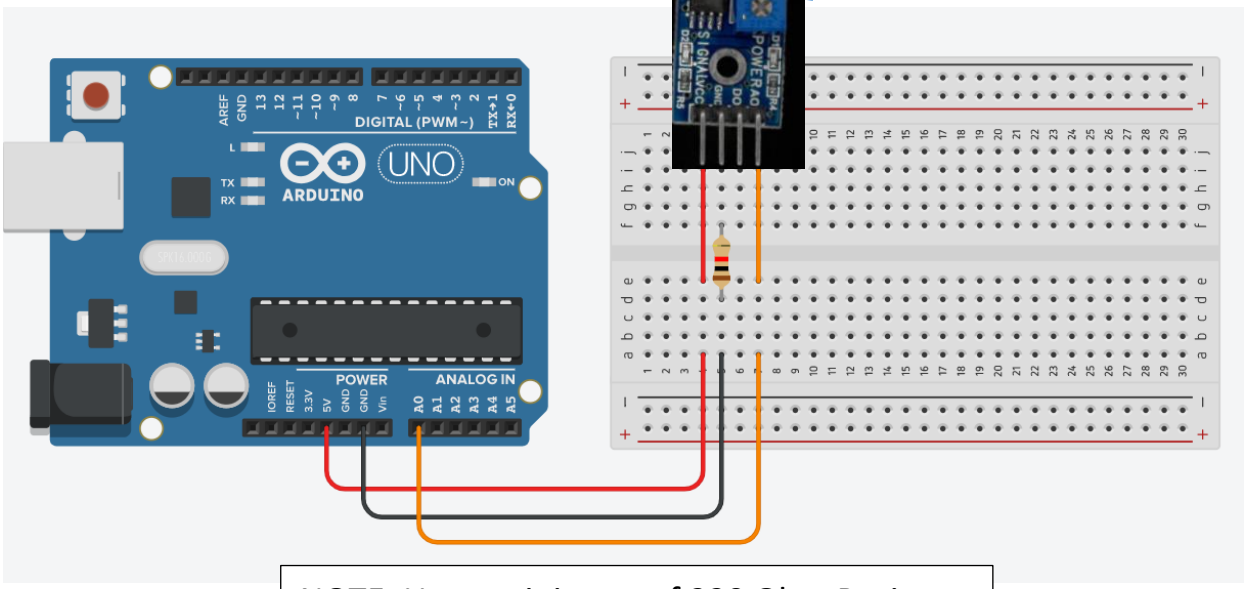
Program 5: Photoresistor Module Sensor

Allows the user to implement a variable resistor that uses light to determine the resistances on the circuit based on the intensity of the light. Can also be used to sense the amount to light intensity around the photoresistor and then can activate other circuits given a defined set of constraints.

Wiring Diagram



NOTE: Use a minimum of 220 Ohm Resistor

Write the following program Testing:

```
int sensorPin = A0;   //sets the photoresistor to Port A0 on the Arduino Board
int value = 0;

void setup()
{
Serial.begin(9600);
}

void loop()
{
value = analogRead (sensorPin);   //Reads the amount of light on the photoresistor
Serial.println(value,DEC);        //Converts the sensor reading to a decimal value
delay(50);                        //Provides Time between Readings
}
```

Upload the program

a. Open the Serial Monitor
b. Place your hand over the Photoresistor > Notice the value change. Analyze the data
c. Close the Serial Monitor > Click on Tools Drop down menu > Select Serial Plotter > now the data is graphed. Analyze the data

Assignment: Simulate Morning, Day and Night Without Looking out the Window

Modify the code that can do the following.

Wire a single RGB LED in place of 3 individual LEDS. See next page for wiring and code (NOTE: code uses a function called SetColor to send values 0-255 for each the Red, Green and Blue Spectrum)

(May consider using a Switch/Case for comparison)
Declare 3 different ranges (I.E 0-100 = Night)

1. Represents Dawn/Dusk = Middle Values
2. Represents Day = Lowest Values (Maximum amount of light)
3. Represents Night: Highest Values (Least amount of light)

NOTE: Adjust the Potentiometer (Blue Box on Sensor Board) to sense larger value ranges

Reference Tutorial RGB LED

Code:

```
1. int redPin = 11;
2. int greenPin = 10;
3. int bluePin = 9;
4.
5. //uncomment this line if using a Common Anode LED
6. //#define COMMON_ANODE
7.
8. void setup()
9. {
10.pinMode(redPin, OUTPUT);
11.pinMode(greenPin, OUTPUT);
12.pinMode(bluePin, OUTPUT);
13.}
14.
15.void loop()
16.{
17.setColor(255, 0, 0); // red
18.delay(1000);
19.setColor(0, 255, 0); // green
20.delay(1000);
21.setColor(0, 0, 255); // blue
22.delay(1000);
23.setColor(255, 255, 0); // yellow
24.delay(1000);
25.setColor(80, 0, 80); // purple
26.delay(1000);
27.setColor(0, 255, 255); // aqua
28.delay(1000);
29.}
30.
31.void setColor(int red, int green, int blue)
32.{
33.analogWrite(redPin, red);
34.analogWrite(greenPin, green);
35.analogWrite(bluePin, blue);
36.}
```

setColor is the name of the function, similar to void setup or void loop. Within the Void Loop setColor is called with 3 Values ranging from 0 to 255 that represent Red (R), Green (G) and Blue (B). These three values are sent to the void setColor Function below and become the values for the integers red, blue, green in that order. The values are then assigned to the specific pin location on the Arduino, mixing the values to create different colors.

Sample code show 6 different color combinations, but more can created by changing the RGB values between 0-255

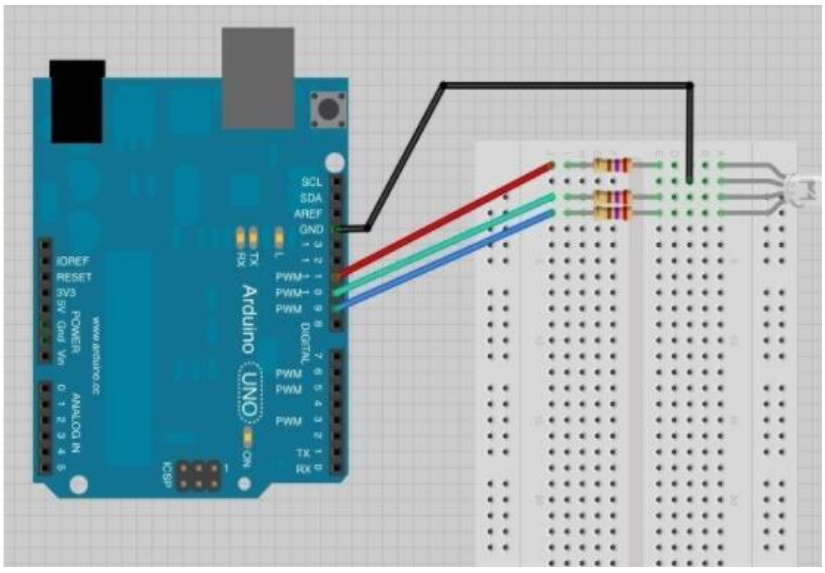
Sample Program to Call a Function

```
void setup() {

inches = ultrasonic (x);
}

long ultrasonic (long microseconds)
{
  Ultrasonic Sensor Code

  return microseconds
}
```

Variable x is passed to Function ultrasonic

Microseconds is returned back to Void Setup where the function was called and overwrite the variable x to what microseconds equals

Wiring Schematic



Note: Color Wires equal the pin color on the RGB LED