Ruth Bader and the Resistor!

AKA

RGB and Photoresistor

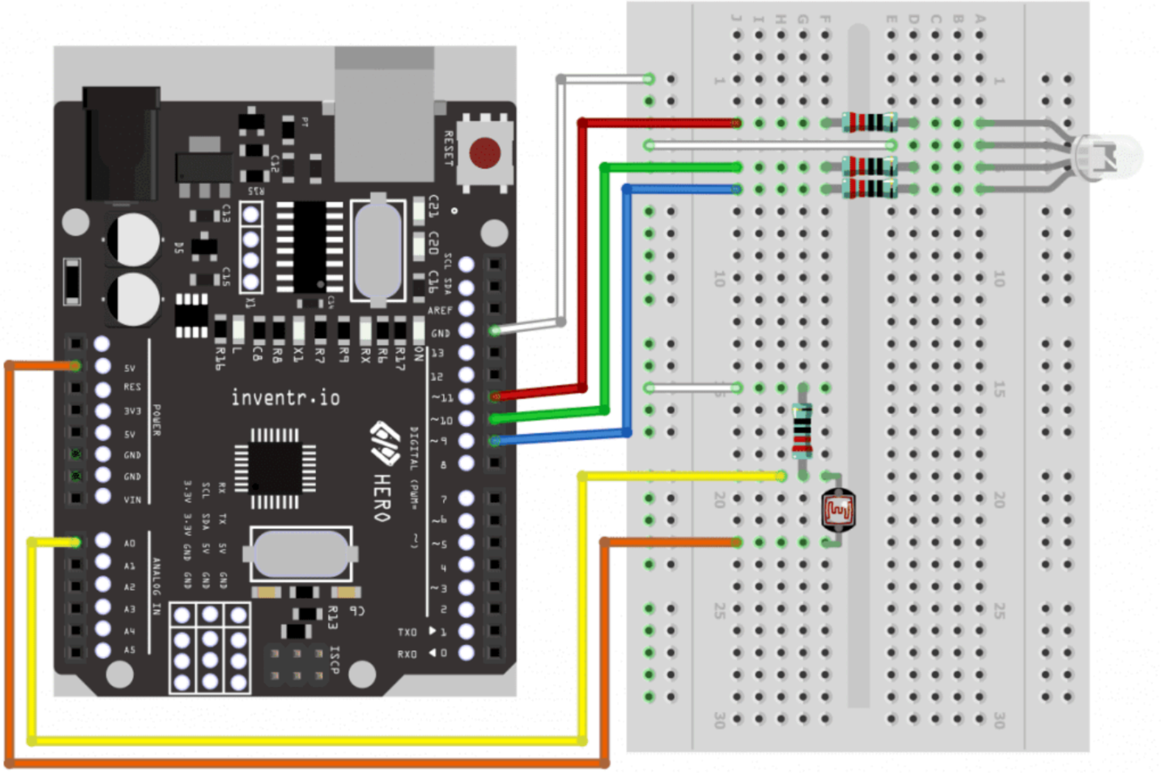
This project will utilize both an RGB led and a Photoresistor. Based on the 'charge', or rather light that is captured by the Photoresistor, both the RGB and the Serial Monitor will reflect a level of 'charge'. This is why the Photoresistor uses the A0 pin, and the RGB led uses the PWM pins. Otherwise, it would not be able to represent a moving number(represented in this project as 'double', or a 'double precision variable').

Pay particular attention to the change in color on the RGB based on the percentage 'full' that the Photoresistor is reading. Based on percentages, the colors will change as the 'percent full' increases.

Please go through each sections to ensure you understand the logic behind each area.

Example code:

```
if (PercentFull > 0 && PercentFull <= 25)
  {
    RGB_color(125, 0, 0); // red
  }
  else if (PercentFull > 25 && PercentFull <= 50){
    RGB_color(125, 80, 0); // yellow-ish
  }
  else if (PercentFull > 50 && PercentFull <= 75) {
    RGB_color(0, 125, 125); // blue-ish
  }
  else if (PercentFull > 75 && PercentFull < 100) {
    RGB_color(125, 125, 125); // white
  }
}
```



Dween Code:

```
int sensorPin = A0;    // select the *analog zero* input pin for probing the photoresistor

int sensorValue = 0;   // variable that we'll use to store the value reading from the sensor

int red = 11;   // these are our output pins for three LED colors

int green = 10;

int blue = 9;


unsigned int batteryCapacity = 50000;
unsigned int batteryLevel = 0;
unsigned int ticks = 0;
unsigned int wait = 100;
double PercentFull;

void setup() {
  // Declare the RGB LED pins as outputs:
  pinMode(red, OUTPUT);
  pinMode(green, OUTPUT);
  pinMode(blue, OUTPUT);

  // Start serial monitor
  Serial.begin(9600);
}
//******************************************

void RGB_color(int red_value, int green_value, int blue_value)
 {
// custom function to set three PWM color channels to any given mixture
// it returns NO numerical result (void)

  analogWrite(red, red_value);
  analogWrite(green, green_value);
  analogWrite(blue, blue_value);
}
//******************************************

 double getBatteryPercentage() {
// custom function calculates battery charge percentage
// and returns the result as a double precision number

  return (((double)batteryLevel / (double)batteryCapacity)*100);
}
//******************************************

void ShowBatteryPercentage() {
  //  send battery results to both serial and LED outputs
   //  calculate the charge percentage with a custom function...
  PercentFull=getBatteryPercentage();

// print the elasped time
  Serial.print(ticks);
  Serial.print(" ms    charge at ");
// print the percent charge
```

```
  Serial.print(PercentFull);
// print a percent character and line return...
  Serial.println("%");

// Now let's light the LED with the appropriate color...

  if (PercentFull > 0 && PercentFull <= 25)
  {
    RGB_color(125, 0, 0); // red
  }
  else if (PercentFull > 25 && PercentFull <= 50){
    RGB_color(125, 80, 0); // yellow-ish
  }
  else if (PercentFull > 50 && PercentFull <= 75) {
    RGB_color(0, 125, 125); // blue-ish
  }
  else if (PercentFull > 75 && PercentFull < 100) {
    RGB_color(125, 125, 125); // white
  }


}

```

```
//******************************************

void loop() {
  sensorValue = analogRead(sensorPin); // read the value from the sensor
  batteryLevel += sensorValue;
  ticks += wait;

  if(batteryLevel >= batteryCapacity) {
    Serial.print(ticks);
    Serial.print(" ms     ");
    Serial.println("FULLY CHARGED");
    batteryLevel = batteryCapacity; // to prevent integer from continuing to increase
    ticks = 0;
    RGB_color(0, 125, 0); // Green for charging complete
    delay(30000);      // long pause
  }
  else {
    ShowBatteryPercentage();
  }

  delay(wait);
}
```