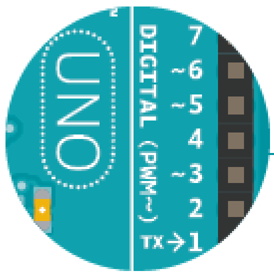
4.Color mixing lamp

Using a tri-color led and three photoresistors, you'll create a lamp that smoothly changes colors depending on external lighting conditions. The Arduino can't vary the output voltage on its pins, it can only output 5V. Hence you'll need to use a technique called **Pulse Width Modulation (PWM)** to fade LEDs.

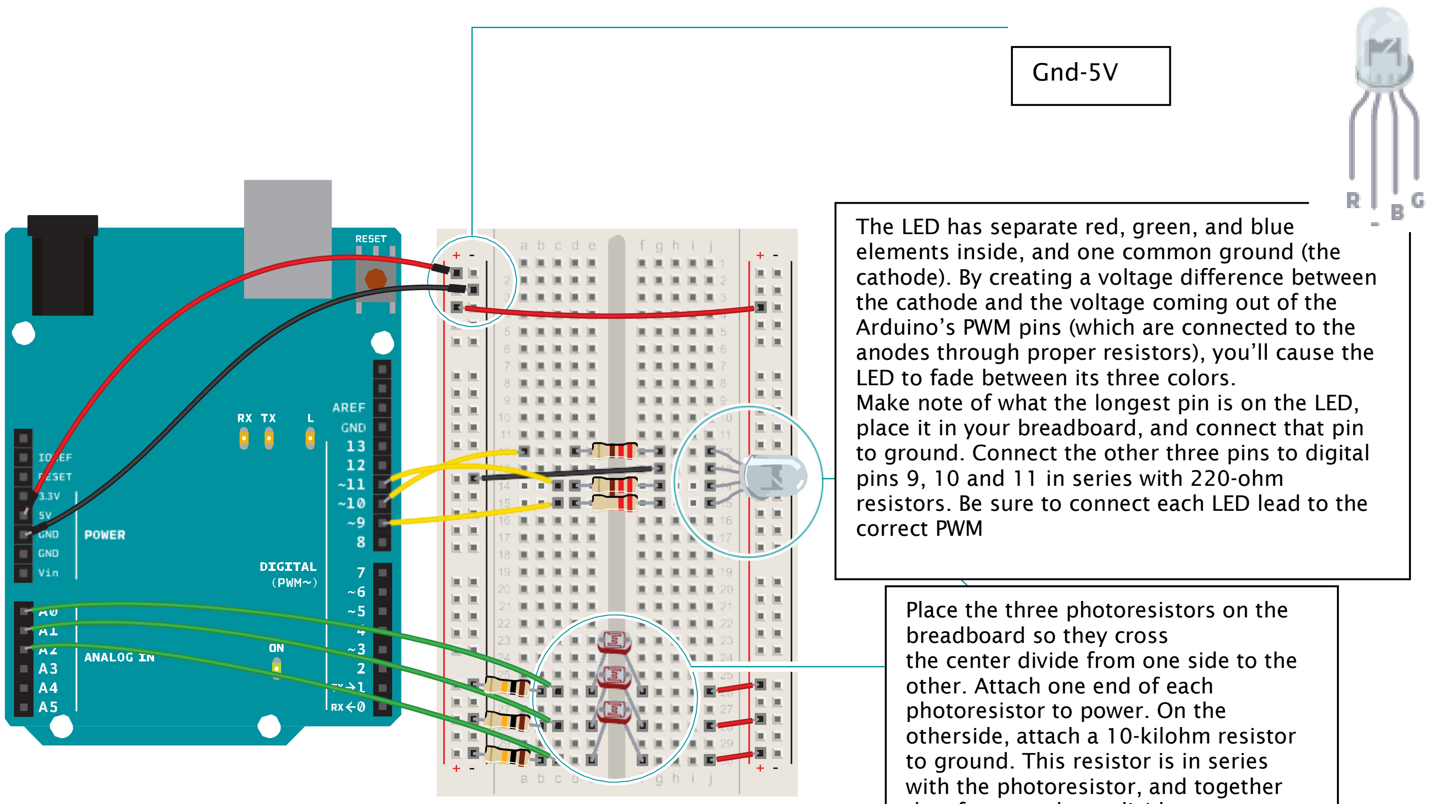
PWM rapidly turns the output pin high and low over a fixed period of time. The change happens faster than the human eye can see. It's similar to the way movies work, quickly flashing a number of still images to create the illusion of motion. When you're rapidly turning the pin HIGH and LOW, it's as if you were changing the voltage. The percentage of time a pin is HIGH in a period is called **duty cycle**. When the pin is HIGH for half of the period and LOW for the other half, the duty cycle is 50%. A lower duty cycle gives you a dimmer LED than a higher duty cycle.

The Arduino Uno has six pins set aside for PWM (digital pins 3, 5, 6, 9, 10, and 11), they can be identified by the ~ next to their number on the board. For inputs in this project, you'll be using photoresistors (sensors that change their resistance depending on the amount of light that hits them, also known as photocells or light-dependent resistors).

If you connect one end of the resistor to your Arduino, you can measure the change in resistance by checking the voltage on the pin.



PWM



Gnd-5V

The LED has separate red, green, and blue elements inside, and one common ground (the cathode). By creating a voltage difference between the cathode and the voltage coming out of the Arduino's PWM pins (which are connected to the anodes through proper resistors), you'll cause the LED to fade between its three colors.
Make note of what the longest pin is on the LED, place it in your breadboard, and connect that pin to ground. Connect the other three pins to digital pins 9, 10 and 11 in series with 220-ohm resistors. Be sure to connect each LED lead to the correct PWM

Place the three photoresistors on the breadboard so they cross the center divide from one side to the other. Attach one end of each photoresistor to power. On the otherside, attach a 10-kilohm resistor to ground. This resistor is in series with the photoresistor, and together they form a voltage divider.
As the resistance of the photoresistor changes when light hits it, the voltage at this junction changes as well. On the same side as the resistor, connect the photoresistors to Analog In pins 0, 1, and 2 with hookup wires.

```
const int greenLEDPin = 9;    // LED connected to digital pin 9
const int redLEDPin = 10;     // LED connected to digital pin 10
const int blueLEDPin = 11;    // LED connected to digital pin 11

const int redSensorPin = A0;  // pin with the photoresistor with the red gel
const int greenSensorPin = A1;   // pin with the photoresistor with the green gel
const int blueSensorPin = A2;   // pin with the photoresistor with the blue gel

int redValue = 0; // value to write to the red LED
int greenValue = 0; // value to write to the green LED
int blueValue = 0; // value to write to the blue LED

int redSensorValue = 0; // variable to hold the value from the red sensor
int greenSensorValue = 0; // variable to hold the value from the green sensor
int blueSensorValue = 0; // variable to hold the value from the blue sensor

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);

  // set the digital pins as outputs
  pinMode(greenLEDPin, OUTPUT);
  pinMode(redLEDPin, OUTPUT);
  pinMode(blueLEDPin, OUTPUT);
}

void loop() {
  // Read the sensors first:

  // read the value from the red-filtered photoresistor:
  redSensorValue = analogRead(redSensorPin);
  // give the ADC a moment to settle
  delay(5);
  // read the value from the green-filtered photoresistor:
  greenSensorValue = analogRead(greenSensorPin);
  // give the ADC a moment to settle
  delay(5);
  // read the value from the blue-filtered photoresistor:
  blueSensorValue = analogRead(blueSensorPin);
```

Set up constants for the pins you're using for input and output, so you can keep track of which sensor pairs with which color on the LED. Use const int for the datatype.
Add variables for the incoming sensor values and for the output values you'll be using to fade the LED. You can use the int datatype for all the variables.
In the **setup(),** begin serial communication at 9600 bps. Just like in the previous example, you will use this to see the values of the sensors in the serial monitor. Additionally, you will be able to see the mapped values you'll use to fade the LED. Also, define the LED pins as outputs with **pinMode().**
In the **loop()** read the sensor values on A0, A1, and A2 with **analogRead()** and store the value in the appropriate variables. Put a small **delay()** between each analogRead() as the ADC takes a millisecond to do its work.
Print out the sensor values on one line.

The “\t” is the equivalent of pressing the “tab” key on the keyboard.

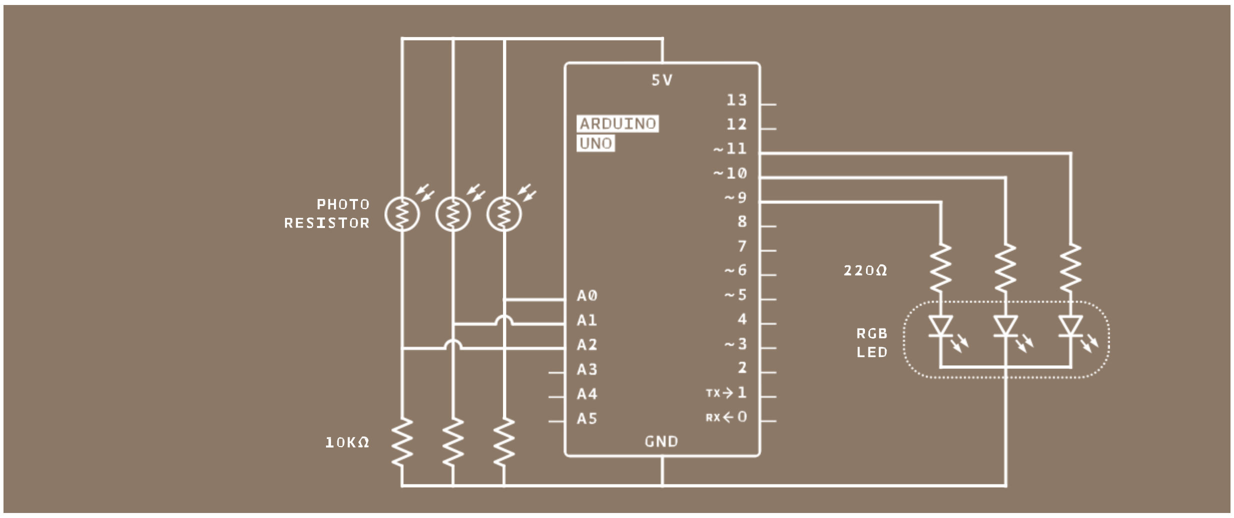
```
  // print out the values to the Serial Monitor
  Serial.print("raw sensor Values \t red: ");
  Serial.print(redSensorValue);
  Serial.print("\t green: ");
  Serial.print(greenSensorValue);
  Serial.print("\t Blue: ");
  Serial.println(blueSensorValue);

  /*
    In order to use the values from the sensor for the LED, you need to do some
    math. The ADC provides a 10-bit number, but analogWrite() uses 8 bits.
    You'll want to divide your sensor readings by 4 to keep them in range
    of the output.
  */
  redValue = redSensorValue / 4;
  greenValue = greenSensorValue / 4;
  blueValue = blueSensorValue / 4;

  // print out the mapped values
  Serial.print("Mapped sensor Values \t red: ");
  Serial.print(redValue);
  Serial.print("\t green: ");
  Serial.print(greenValue);
  Serial.print("\t Blue: ");
  Serial.println(blueValue);

  /*
    Now that you have a usable value, it's time to PWM the LED.
  */
  analogWrite(redLEDPin, redValue);
  analogWrite(greenLEDPin, greenValue);
  analogWrite(blueLEDPin, blueValue);
}
```

The function to change the LED’s brightness via PWM is called **analogWrite()**. It needs two arguments: the pin to write to, and a value between 0-255. This second number represents the duty cycle the Arduino will output on the specified pin. A value of 255 will set the pin HIGH all the time, making the attached LED as bright as it can be. A value of 127 will set the pin HIGH half the time of the period, making the LED dimmer. 0 would set the pin LOW all the time, turning the LED off. To convert the sensor reading from a value between 0-1023 to a value between 0-255 for analogWrite(), divide the sensor reading by 4.
Print out the new mapped values on their own line.


5V
ARDUINO
UNO
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
A0
A1
A2
A3
A4
A5
GND
PHOTO
RESISTOR
10KΩ
220Ω
RGB
LED