Introduction:
So far we have used normal LEDs, which come in a variety of colours, but now we are going to use an RGB LED. These are capable of displaying any colour light within the RGB spectrum. You are going to learn how to control it’s colour output, first using a button and random function, and then with a potentiometer.

Goals
• Understand the principles of RGB light.
• Program an LED to display a random colour.
• Control an RGB using a potentiometer and Serial Communications.
The RGB LED

RGB stands for Red Green and Blue, which make up our additive primary colours. When these colours are mixed, they can be used to create a variety of colours.

This LED, unlike the others, have 4 Legs instead of 2. This is because, Inside this LED is effectively three LEDs, one Red, one Green and one Blue, with a common cathode leg. This means that all three colours use the same leg that goes to ground to complete the circuit.

Unlike normal LEDs, where the longer leg represents the anode leg, in this RGB LED the longer leg is actually the cathode. This means our electricity will flow through one of the shorter legs, and out the longer leg to ground.

05 - RGB LED Challenge

Why does an RGB LED have 4 legs?

How many colours do you think an RGB LED can make?
Let’s make our RGB led light up!

For this, we will need:

1x Arduino Uno
1x USB Cable
1x RGB LED
3x 1Kilo Ohm Resistor
(Brown Black Red Gold)
6x Jumper Wires
Make a new sketch in the Arduino IDE and name this ‘RGB_Random_light’.

```cpp
//rgb random light
//set our pins for each colour
int redPin = 11;
int bluePin = 9;
int greenPin = 10;

//variables to store the colour brightness
int red = 255;
int blue = 255;
int green = 255;

void setup() {
  //put your setup code here to run once;
  //set each pin to OUTPUT
  pinMode(redPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
  pinMode(greenPin, OUTPUT);

  //set up pins initially as fully bright
  analogWrite(redPin, red);
  analogWrite(bluePin, blue);
  analogWrite(greenPin, green);

  //pause setup for 3000 milliseconds so the light will stay on one colour
  delay(3000);
}

void loop() {
  //put your main code here to run repeatedly:
  //populate each number with a random number between 0 and 255
  red = random(0, 255);
  blue = random(0, 255);
  green = random(0, 255);

  //use this random number to set the brightness of each colour
  analogWrite(redPin, red);
  analogWrite(bluePin, blue);
  analogWrite(greenPin, green);

  //delay this for 500 milliseconds (1/2 a second)
  delay(500);
}
```
Once you have copied the code, press (compile) and if no errors appear, press (upload) and watch the results!

My code won't compile!

Is everything spelt correctly?
Are all your lines ending in a semi-colon?
Do you have the correct capital letters?
Did you close the curly brackets?

What can you see happening?

You should notice the RGB LED displaying a different colour every 500 milliseconds, all made up of Red Green and Blue. The difference in colour is achieved by fading each LED using analogWrite(). This is how we use PWM.

Let’s go through the code understand what every part is doing.

Variables

These are our variables set with the values of each different colours anode on the RGB LED. This means it’s easier for us to remember which pins are set for controlling each colour.

```c
int redPin = 11;
int bluePin = 9;
int greenPin = 10;
```

Global Variables

These global variables are for storing each colours brightness value. They are all set to 255 so they start at full brightness.

```c
int red = 255;
int blue = 255;
int green = 255;
```

void setup()

This is our void setup function. Everything within the curly brackets will be called only once at the start of our program.

```c
void setup() {

```

pinMode()

This is how we set our digital pins we are using to control the LEDs as outputs. This is called in void setup as it only needs to be called once.

```c
pinMode(redPin, OUTPUT);
pinMode(bluePin, OUTPUT);
pinMode(greenPin, OUTPUT);
```
This delay function means our void setup function is paused for 3000 milliseconds before it completes. This means that on start up, our project will pause showing all colours in the RGB LED at full brightness.

We then pause again so that the created colour stays the same for 500 milliseconds.

This is how we set each of our colours brightness. We use Arduino function random() to generate a number between 0 and 255 for each colour. These are then stored in the global variables for initialized at the top of our program.

Once we’ve generated our random values, we then use analogWrite() to turn the values into brightness for each LED colour.
Now your challenge is to further your RGB LED Sketch and create a controllable RGB LED. The basic circuit will stay the same, but you will need to create an additional circuit for the potentiometer. The code will need to use Serial communication.

We have not included any further information, as we think you know it already! Look back through the last few exercises if you need a reminder.

Can you create a robot that has a RGB LED and a Servo Motor both controlled by a potentiometer?

In the Servo pack in your Start Arduino kit, there should be different arms and pulleys. Experiment with these and see what you can create with them. Try making an arm out of card and stick it to your servo pulley. Have fun!