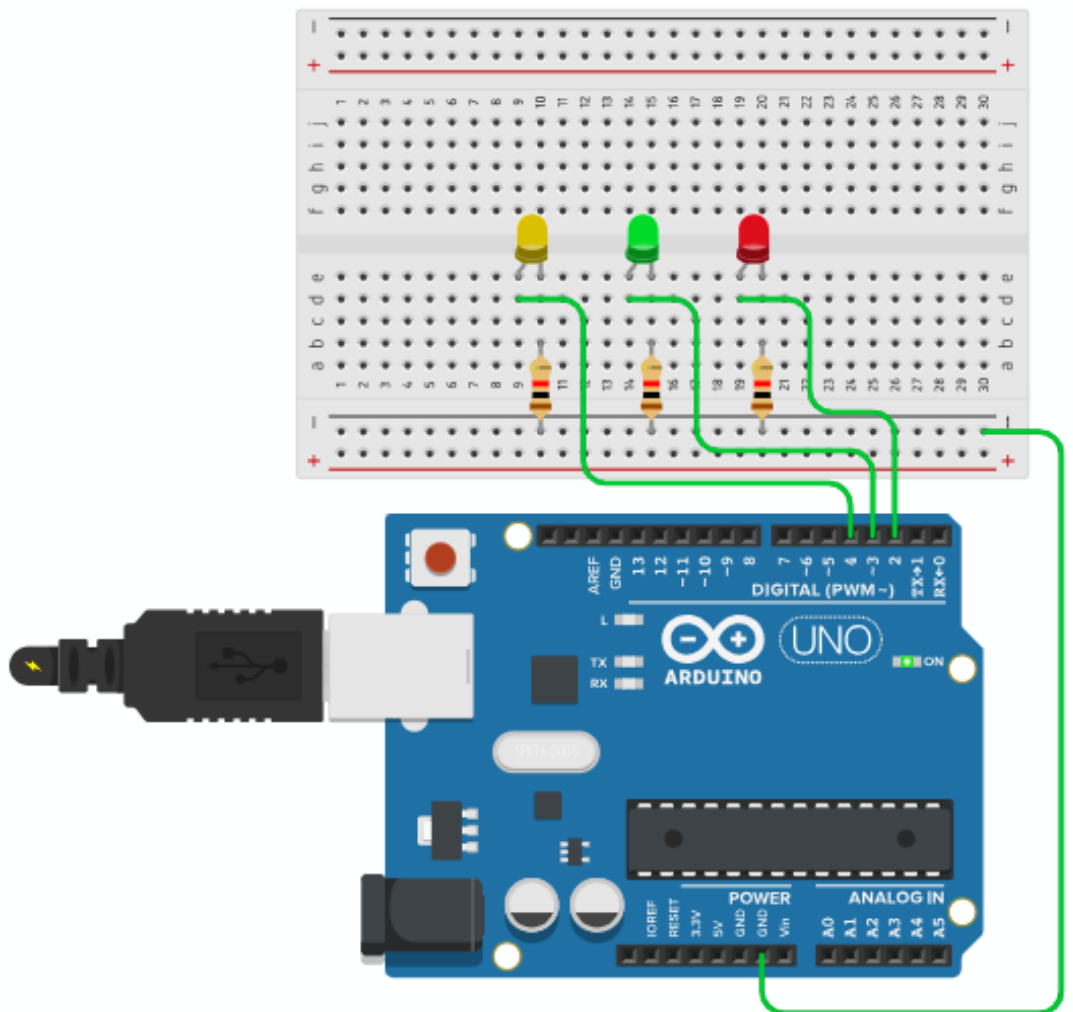


Activity: Write an Arduino Program to Make 3 Color LED Patterns

Objective:

Ever noticed a level indicator? Whether it is a battery level indicator, voltage level, water level, speed level or distance level, they all have multiple colored LEDs to depict different levels.

Although the sequence of the colors may vary, they all have at least three LEDs depicting Low, Medium and High level.











So, we can say without a doubt that we come across such multi color multi level LED indicators almost everywhere in our day to day life. Now, wouldn't it be interesting if we are able to devise our own 3 LED 3 color arrangement and make them run many different display patterns through blinking ?

So, in today's activity we will write an Arduino program that will control 3 LEDs of different colors. The setup will generate different blinking patterns using all the 3 LEDs.

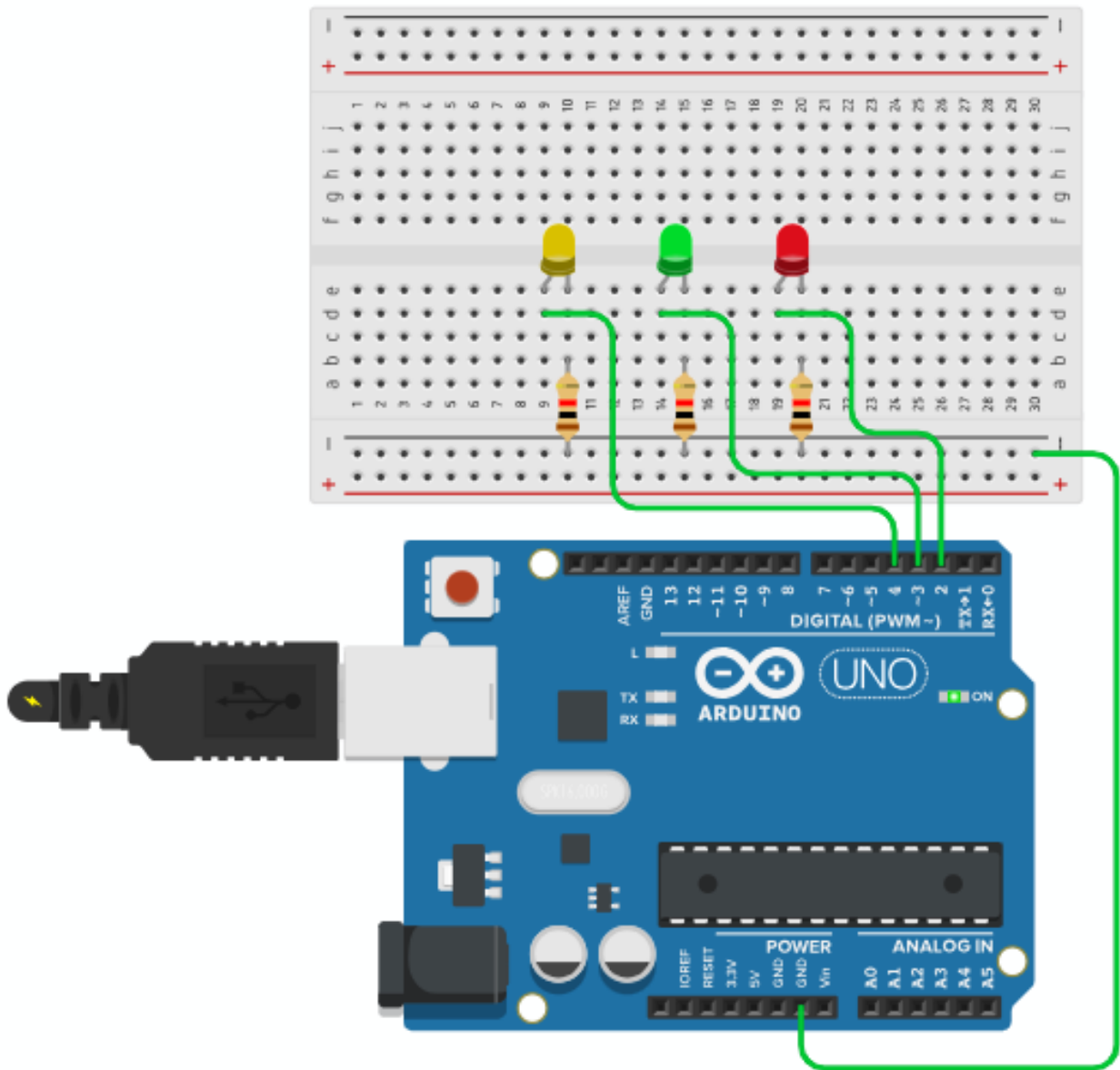
All Arduino boards have an on-board LED light connected to pin D13. An external LED, on the other hand, can be connected to any digital pin of the arduino.

Materials Required:

| S.no. | Name | Qty | Image |
|-------|--------------------|-----|---|
| 1 | Arduino | 1 |  A blue Arduino Uno R3 microcontroller board. It features a USB Type-B port, a DC power jack, a reset button, and various digital and power pins. The board is labeled 'ARDUINO UNO R3' and 'DIGITAL 5V GND'. |
| 2 | Breadboard | 1 |  A standard white breadboard used for prototyping electronic circuits. It has a grid of holes and a central channel for ICs. |
| 3 | M - M jumper wires | 10 |  A bundle of 10 multi-colored jumper wires. Each wire has a different color (red, yellow, green, blue, black) and is terminated with a female header connector on one end and a male header connector on the other. |

| | | | |
|---|------------------------------------|---|---|
| 5 | USB cable for Arduino | 1 |  |
| 6 | Resistors: 1k (Brown Black Red) | 3 |  |
| 7 | Red LED | 1 |  |
| 8 | Green LED | 1 |  |
| 9 | Yellow LED | 1 |  |

Connection Diagram:



Explanation:

Although the above circuit looks seemingly complicated, it is quite simple if you take one step at a time.

1. Place the 3 LED on the breadboard such that
 - a. The +ve lead is on the left and -ve lead is on the right.
 - b. Leave a gap of 2 points between each LED.
2. Take three 1k resistors and insert them right below the -ve lead of the LEDs as shown in the image above.

3. Take a M-M jumper wire and insert it between the three 1k resistors connected to the Gnd line (shown as black line on the breadboard) and the Arduino's Gnd pin.
4. Take 3 M-M wires and connect them between the +ve lead of the LEDs and the digital pins 2, 3 and 4 of the Arduino on the breadboard. These are the three green wires coming out from the pins D2, D3 and D4 of the Arduino and going all the way to the points right under the +ve leads of the LEDs.
 - a. D2 is connected to Red's +ve.
 - b. D3 is connected to Green's +ve.
 - c. D4 is connected to Yellow's +ve.

Arduino Code:

The program for this activity is different from the previous program in many respects. Here, we introduce you to the concept of functions, controlling multiple outputs (LEDs), using human readable names for different arduino pin numbers.

So far we have seen only 2 functions which are the default (predefined) functions within the Arduino environment:

1. The "setup" function and
2. The "loop" function.

As already known by now, "setup" is used for things we want to do just once like declaring a particular pin as Output while "loop" is used for things that we want to keep doing repetitively such as turning a pin high and low (for turning an LED on and off).

Any function other than these two are called user defined functions and a lot about them is going to be clear after going through the following code. Additionally, instead of denoting pins as mere numbers like 2, 3 and 4 we will now start to name them as more human readable forms such as Red, Green and Yellow since these are the pins associated with these colored LEDs.

This activity will also introduce you to "for" loops. Loops are a way of doing a part of the code repetitively for a defined number of times. For instance, let's say you want

the Red LED to blink 5 times, the Green LED 3 times and the Yellow LED just once; you can achieve this using a “for” loop. Here is the complete code.

```
#define red 2 // defining pin 2 as "red"
#define grn 3 // defining pin 3 as "grn"
#define ylw 4 // defining pin 4 as "ylw"

void setup()
{
  pinMode(red, OUTPUT); // declaring "red" pin as output
  pinMode(grn, OUTPUT); // declaring "grn" pin as output
  pinMode(ylw, OUTPUT); // declaring "ylw" pin as output
}

void loop()
{
  for(int i=0;i<5;i++)
  {
    pattern1();
  }

  for(int i=0;i<5;i++)
  {
    pattern2();
  }

  for(int i=0;i<5;i++)
  {
    pattern3();
  }

  for(int i=0;i<5;i++)
  {
    pattern4();
  }
}

void pattern1()
{
```

```
digitalWrite(red, HIGH); // turn the red LED ON
delay(400); // wait for 1 second
digitalWrite(red, LOW); // turn the red LED OFF

digitalWrite(grn, HIGH); // turn the grn LED ON
delay(400); // wait for 1 second
digitalWrite(grn, LOW); // turn the grn LED OFF

digitalWrite(ylw, HIGH); // turn the ylw LED ON
delay(400); // wait for 1 second
digitalWrite(ylw, LOW); // turn the ylw LED OFF
}
```

```
void pattern2()
```

```
{
digitalWrite(ylw, HIGH); // turn the ylw LED ON
delay(400); // wait for 0.4 second
digitalWrite(ylw, LOW); // turn the ylw LED OFF

digitalWrite(grn, HIGH); // turn the grn LED ON
delay(400); // wait for 0.4 second
digitalWrite(grn, LOW); // turn the grn LED OFF

digitalWrite(red, HIGH); // turn the red LED ON
delay(400); // wait for 0.4 second
digitalWrite(red, LOW); // turn the red LED OFF
}
```

```
void pattern3()
```

```
{
digitalWrite(grn, HIGH); // turn the grn LED ON
digitalWrite(red, LOW); // turn the red LED OFF
digitalWrite(ylw, LOW); // turn the ylw LED OFF

delay(500); // wait for half second

digitalWrite(grn, LOW); // turn the grn LED OFF
digitalWrite(red, HIGH); // turn the red LED ON
digitalWrite(ylw, HIGH); // turn the ylw LED ON
}
```

```

    delay(500);           // wait for half second
}

void pattern4()
{
    digitalWrite(red, HIGH); // turn the red LED ON
    digitalWrite(grn, HIGH); // turn the grn LED ON
    digitalWrite(ylw, HIGH); // turn the ylw LED ON
    delay(1000);           // wait for 1 second
    digitalWrite(red, LOW); // turn the red LED OFF
    digitalWrite(grn, LOW); // turn the grn LED OFF
    digitalWrite(ylw, LOW); // turn the ylw LED OFF
    delay(1000);           // wait for 1 second
}

```

Explanation:

The code is divided into three sections.

1. "setup"
2. "loop" and
3. LED patterns functions: pattern1, pattern2, patter3, pattern4.

The "setup" section is used for events or actions that are done only once, while the "loop" section is used for events and actions that are done repetitively forever (till the power is ON).

Declaring which pin will work as output is needed just once and is therefore written inside the "setup" section. The turning ON and OFF of the LED, on the other hand, needs to be done repetitively and so it is written inside the "loop" section.

```
#define red 2           // defining pin 2 as "red"
```

This statement is defining that from now on we will call the pin D2 as "red". Similarly we use "grn" and "ylw" for the pins D3 and D4.

```
pinMode(red, OUTPUT); // declaring "red" pin as output
```

This statement is declaring the "red" pin (which is pin D2) as an Output pin. Similarly we use statements that declare the pins "grn" and "ylw" also as output.


```
for(int i=0;i<5;i++)
{
    pattern1();
}
```

This set of statements forms a “for” loop. A “for” loop has a working scope denoted by the curly braces “{” and “}”. All statements within this set of curly braces are affected by the “for” loop.

A “for” loop has 3 parts:

1. Initial counter value with condition.
2. Final counter value with condition and
3. Change in counter value i.e. increment or decrement with condition.

All these parts are inside the “(” and “)” small bracket pair that immediately follows the word “for”.

```
for(int i=0;i<5;i++)
```

In this statement:

1. Initial counter value is the integer variable “i” with the condition that it should be equal to 0 (zero).
2. Final counter value is the same integer variable “i” with the condition that it should be less than 5.
3. Change in counter value is incremented by 1 denoted by “i + +”. Had it been decremented by 1, it would be denoted by “i - -”.
4. This entire setup simply means:
 - a. For the range of 0 (zero) to less than 5 i.e. for the counter values of 0,1,2,3 and 4, keep doing whatever is within the curly braces “{” and “}”.
 - b. A more understandable approach is “**do something for 5 times**” and that **something** is mentioned between the curly braces.

```
for(int i=0;i<5;i++)
{
    pattern1();
}
```

Since we have written the function “pattern1” within the curly braces, it gets repeated 5 times. Similarly, for all the other display patterns we have used the same technique for running them for 5 times and then moved on to a different pattern.

So, in these “for” loops, if you change the range of the counter variable “i”, you will run them a different number of times. Here is the difference.

```
for(int i=0;i<5;i++)
{
    pattern1();
}
```

This will run pattern 1 **5 times**

```
for(int i=1;i<11;i++)
{
    pattern1();
}
```

This will run pattern 1 **10 times**

```
for(int i=10;i>0;i--)
{
    pattern1();
}
```

This will also run pattern 1 **10 times**

Lastly, the basic statements for the LED control are the following.

```
digitalWrite(red, HIGH); // turn the red LED ON
delay(400); // wait for 1 second
digitalWrite(red, LOW); // turn the red LED OFF
```

The first statement digitally writes the “red” pin (which is actually the D2 pin) with logical HIGH (which is 5v). Thus a HIGH on this pin turns the connected LED On.

The third statement digitally writes the “red” pin (which is actually the D2 pin) with logical LOW (which is 0v). Thus a LOW on this pin turns the connected LED Off.

The second statement is simply a delay for 400 milliseconds. The Arduino will halt its execution for this designated amount of time and will resume when this time has passed.

So, throughout the length of the entire program, wherever we do a digitalWrite with HIGH, we are turning the LED On and wherever we do a digitalWrite with LOW, we are turning the LED Off. And wherever we use delay with a number, the LED **stays** On or Off for that duration.

Thus, we see that no matter what pattern we display, how many LEDs we use, or how the patterns appear in sequence, all we are doing is one of the 2 following things.

1. An Event (turning the LED On or Off).
2. A Duration (time for which the LED stays On or Off).

Looking at any arduino code in this way simplifies the approach a lot. It also gives us the confidence that a large problem can ultimately be broken down into its constituents and these can then be handled with far greater ease than the problem as a whole.

Outcome and Observations:

1. Once you are done uploading the code, all the LED patterns start to play one by one.
2. The first pattern is such that the LEDs will glow from Left to Right i.e. first the Red LED will turn On while the other two remain Off. Next, the Green LED will turn On while Red and Yellow LEDs remain Off and lastly the Yellow LED turns On while the Red and Green LEDs remain Off. This creates an illusion of the light pattern moving from Red to Yellow i.e. from Left to Right.
3. The second pattern is such that the LEDs will glow from Right to Left i.e. first the Yellow LED will turn On while the other two remain Off. Next, the Green LED will turn On while Red and Yellow LEDs remain Off and lastly the Red LED turns On while the Yellow and Green LEDs remain Off. This creates an illusion of the light pattern moving from Yellow to Red i.e. from Right to Left.
4. The third pattern is such that first the Green LED turns On while the other two remain Off and then the Red and Yellow LEDs turn On while the Green LED remains Off thus giving an alternate blinking pattern. When the middle LED is On the side ones are Off and when the side ones go On the middle one goes Off.

5. The fourth pattern is such that all the 3 color LEDs turn On together and the turn Off together as well thus producing an altogether blinking pattern.
6. Every pattern has delay functions, which, when changed, will change the On time and Off time of the LEDs and thus change the overall appearances of the patterns.
7. Starting from the first to last:
 - a. Pattern 1 runs 5 times and then shifts to Pattern 2.
 - b. Pattern 2 runs 5 times and then shifts to Pattern 3.
 - c. Pattern 3 runs 5 times and then shifts to Pattern 4.
 - d. Pattern 4 runs 5 times and then goes back to Pattern 1.
8. To make the individual blinking patterns stand out, you can turn all LEDs off after the pattern and add a long delay in between the “for” loops. Figure this part out on your own. Happy brainstorming.