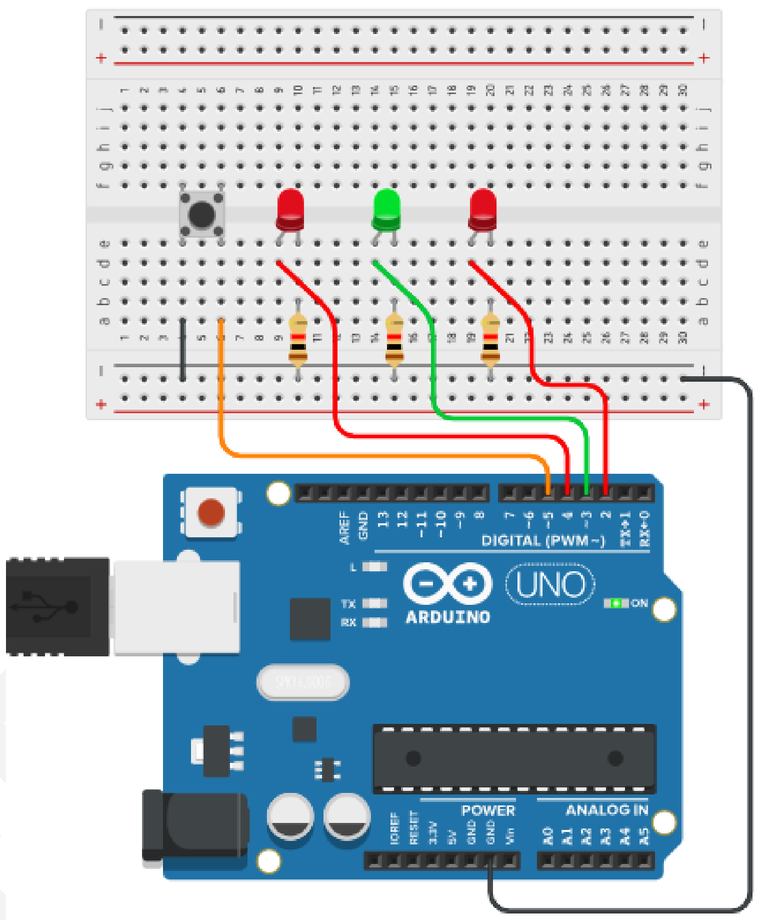
Activity: Write a Program to Control LEDs using a Push Button

Objective:



So far in our activities we have already controlled single as well as multiple LEDs with the Arduino program but LEDs are only output devices and we have already seen how to control outputs using “digitalWrite”.

But digital systems don't just have Outputs, they have Inputs as well and so we need to learn how to interface input devices with the Arduino. One of the most basic input devices is a simple Push button Switch. A push button switch is a simple make-contact and break-contact type of input device that can be used to receive inputs. As we all already know, a push button switch can either be in a “pressed” state or a “released” state. When the button is pressed, the contact is made and when the button is released, the contact is broken.

So, in today’s activity we will first learn how to connect a Push button switch to the Arduino, then we will learn how to control an LED using the “press” and “release” operation of the button and finally, how to use the button to increment a counter value and display the output on multiple LEDs. This activity will be divided into 2 parts.

1. Detecting when the button is pressed and when it is released.
2. Producing different LED patterns based on a counter value which will be changed every time the button is pressed.

E.g. Initially all LEDs will be Off. pressing the button once will display pattern 1, pressing it again will display pattern 2 and pressing it again will show pattern 3 and so on.

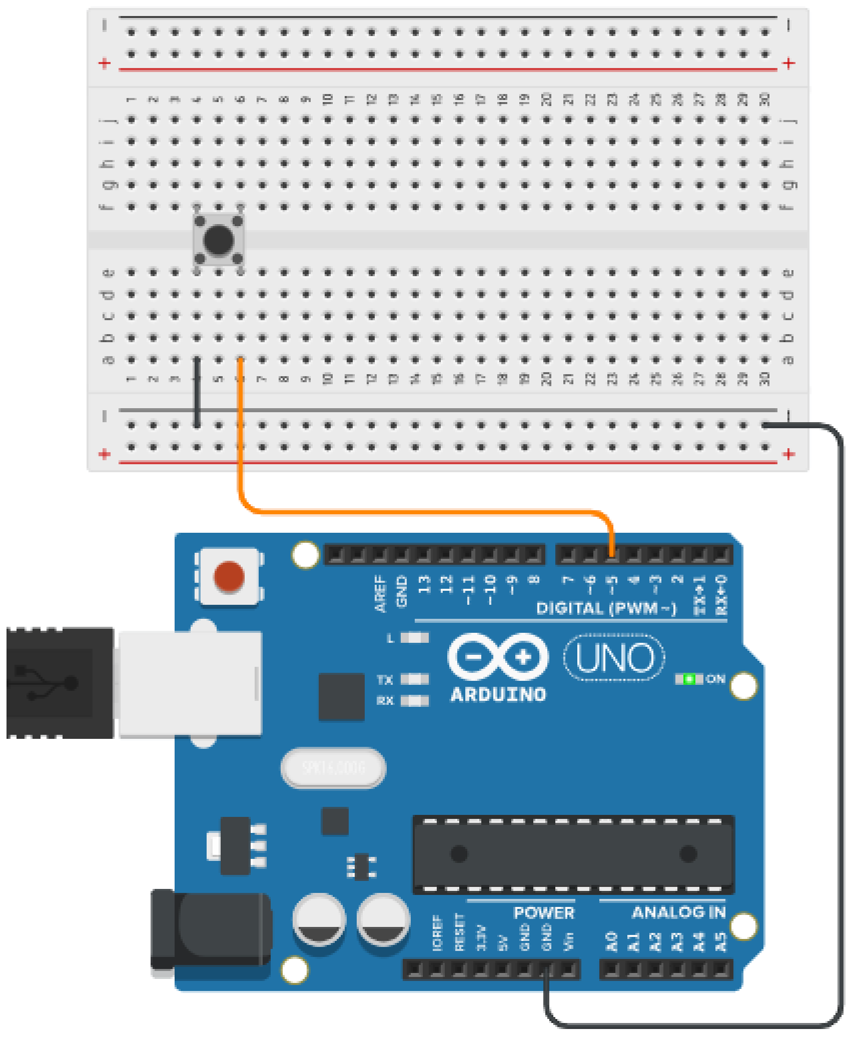
Materials Required:

S.no.	Name	Qty	Image
1	Arduino Nano	1	
2	Breadboard	1	
3	M - M jumper wires	15	
5	USB cable for Arduino Nano	1	

6	Resistors: 1k (Brown Black Red)	3	
7	Red LED	2	
8	Green LED	1	
9	4-pin tact switch	1	

Part 1:

Connection Diagram:



Explanation:

Looking at the connection diagram shown above, some things are immediately evident:

1. One side of the Push button switch is connected to Digital Pin 5 (D5) of the Arduino while the other side is connected to -ve of the Breadboard with a connection wire (shown with the black line just below the switch).
2. The +ve and -ve power pins of the Arduino are connected to the +ve and -ve lines of the Breadboard.
3. Pressing the Button connects Pin 5 (D5) of the Arduino to Gnd (-ve power) and releasing the button leaves the Pin 5 unconnected.

Since we only want to see whether the button has been pressed or not without any additional hardware and connections, we will use the on-board LED of the Arduino which is connected to Digital Pin 13 (D13). Pressing the Button should turn the LED On and releasing the Button should turn the LED Off.

Arduino Code:

So far we have only controlled Outputs by using the “digitalWrite(pin, state)” function but Inputs are different. We don't turn Inputs On or Off, we check them if they are On or Off. So essentially we are doing the opposite of a “digitalWrite” and so taking an Input is referred to as Reading and the associated function is a “digitalRead” function.

Since we cannot specify a High or Low inside the digitalRead function, we only specify the Pin which we wish to Read. A typical read of the input is written as shown in the statement below.

```
digitalRead(2);  // get status of pin 2 of Arduino
```

Similarly there will be changes in the Setup() function as well because now we need to declare which pin we intend to use as Input using the pinMode() function.

This will be more clear when you go through the code below.

Here is the Arduino code.

```
#define LED 13                 // defining pin 13 as "LED"
#define Button 5               // defining pin 5 as "Button"

int Button_state;

void setup()
{
  pinMode(LED, OUTPUT);          //declaring "LED" as output
    pinMode(Button,  INPUT_PULLUP);  //declaring  "Button"  as
input
}

void loop()
```

```
{
  Button_state = digitalRead(Button);
  digitalWrite(LED, !Button_state);
}
```

Explanation:

```
#define LED 13                 // defining pin 13 as "LED"
#define Button 5               // defining pin 5 as "Button"
```

The first two lines are simply defining which pins are going to be used as what. This makes it easier for the coder or user to refer to the pins in a more human understandable form. So, instead of referring to them using numbers, we give them names such as LED and Button.

```
int Button_state;
```

In this line we are declaring that we are going to use an Integer type variable with the name “Button_state”. A variable can be of many types depending on the value we want it to hold. It can be an Integer (number without a decimal point), a Character (a-z or A-Z), a String (set of alphabets e.g. “robot” or “ROBOT” or “Robot”) etc.

```
void setup()
{
  pinMode(LED, OUTPUT);          //declaring "LED" as output
    pinMode(Button,  INPUT_PULLUP);  //declaring  "Button"  as
input
}
```

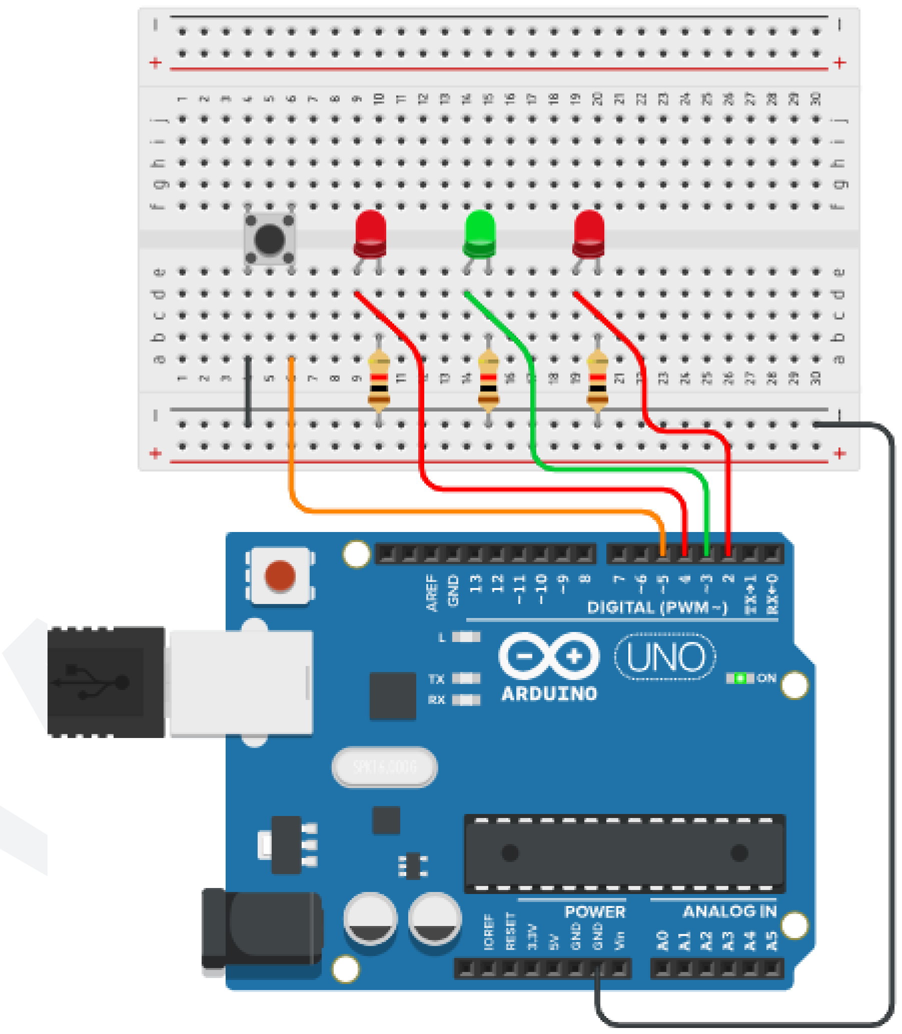
Here we are declaring that the “LED” will be used as an Output and that the “Button” will be used as an Input. Note that, here, we are using “INPUT_PULLUP” instead of just “INPUT”. This is because Pressing the Button makes D5 LOW but Releasing it will not make D5 HIGH but we do need this HIGH state by default. And so we are activating the Arduino’s Internal Pull-Ups by using “INPUT_PULLUP” instead of just “INPUT”.

```
void loop()
{
  Button_state = digitalRead(Button);
  digitalWrite(LED, !Button_state);
}
```

Here, we are first reading the state of the Push Button and storing it in the variable “Button_state”. Pressed is stored as a 0 and Released is stored as a 1. And the we are turning the LED On or Off by writing this 0 or 1 on the “LED” pin i.e. D13.

Part 2:

Connection Diagram:



Explanation:

In the second part of the activity, we use 3 external LEDs, 2 x Reds and 1 x Green and each is connected to a different pin on the Arduino. Take a look at the above connection diagram and you will observe the following:

1. The First LED is connected to the Arduino’s digital pin 4 (D4) such that the +ve pin of the LED is connected to D4 and the -ve pin of the LED is connected to a 1k (brown, black,red) resistor. While the other side of this resistor is connected to the Gnd line of the breadboard.
2. The Second LED is connected to the Arduino’s digital pin 3 (D3) such that the +ve pin of the LED is connected to D3 and the -ve pin of the LED is connected to a 1k (brown, black,red) resistor. While the other side of this resistor is connected to the Gnd line of the breadboard.
3. The Third LED is connected to the Arduino’s digital pin 2 (D2) such that the +ve pin of the LED is connected to D2 and the -ve pin of the LED is connected to a 1k (brown, black,red) resistor. While the other side of this resistor is connected to the Gnd line of the breadboard.
4. The push button is connected such that its one side goes to D5 of Arduino (shown with orange line) and other side to the Gnd of the breadboard (shown with black line).

Arduino Code:

Now that we have already understood how a push button works for a single LED, we will move on to control 3 LEDs one by one . Here is the code for controlling 3 LEDs with a push button switch.

```
#define LED1 4                // defining pin 4 as "LED1"
#define LED2 3                // defining pin 3 as "LED2"
#define LED3 2                // defining pin 2 as "LED3"

#define Button 5              // defining pin 5 as "Button"

int Button_state;
int Counter=0;


void setup()
{
  pinMode(LED1, OUTPUT);      // declaring "LED1" as output
  pinMode(LED2, OUTPUT);      // declaring "LED2" as output
  pinMode(LED3, OUTPUT);      // declaring "LED3" as output
  pinMode(Button, INPUT_PULLUP); // declaring "Button" as output
}


void loop()
{
  Button_state=digitalRead(Button);   // check if button is pressed

  if(Button_state == LOW)         // if button is in pressed state
  {
    Counter = Counter + 1;        // increment Counter by 1
    delay(500);                   // wait for half a second
  }

  if(Counter == 1)                // if Counter value is 1
  {
    digitalWrite(LED1,HIGH);// turn On first LED
  }

  if(Counter == 2)                // if Counter value is 2
  {
    digitalWrite(LED2,HIGH);// turn On second LED
  }

  if(Counter == 3)                // if Counter value is 3
  {
    digitalWrite(LED3,HIGH);// turn On third LED
  }

  if(Counter == 4)                // if Counter value is 4
  {
    digitalWrite(LED1,LOW);       // turn Off first LED
    digitalWrite(LED2,LOW);       // turn Off second LED
    digitalWrite(LED3,LOW);       // turn Off third LED
    Counter = 0;                  // reset Counter to 0
  }
}
```

Explanation:

The above code does the following:

1. Check whether the push button is pressed or not.
2. If the button is pressed, increment a Counter value by 1.
3. If the Counter value is 1, turn On the 1st LED.
4. If the Counter value is 2, turn On the 2nd LED.
5. If the Counter value is 3, turn On the 3rd LED.
6. If the Counter value is 4, turn Off all the LEDs and Reset the Counter to 0.

```
#define LED1 4               // defining pin 4 as "LED1"
#define LED2 3               // defining pin 3 as "LED2"
#define LED3 2               // defining pin 2 as "LED3"

#define Button 5             // defining pin 5 as "Button"
```

These lines are defining that the 1st, the 2nd and the 3rd LEDs are connected to Arduino Digital pins 4, 3 and 2 respectively and the button is connected to Arduino Digital pin 5.

```
int Button_state;
int Counter=0;
```

Here, we have taken 2 variables, both integer types, one for the button and the other for storing the Counter value. The Counter value has been initialized by 0.

```
void setup()
{
  pinMode(LED1, OUTPUT);     // declaring "LED1" as output
  pinMode(LED2, OUTPUT);     // declaring "LED2" as output
  pinMode(LED3, OUTPUT);     // declaring "LED3" as output

  pinMode(Button, INPUT_PULLUP); // declaring "Button" as output
}
```

Here, in the setup function, we are declaring the pins used for the LED as Outputs and the one used for the button is declared as Input with its internal pull-up activated.

```
 Button_state=digitalRead(Button);    // check if button is pressed
```

This is the line which actually checks the state of the button whether it is in the pressed state or released state and then stores that state in the “Button_state” variable. The way we have connected our push button, “digitalRead” of a pressed state will be 0 (LOW) while “digitalRead” of a released state will be 1 (HIGH).

```
if(Button_state == LOW)    // if button is in pressed state
{
  Counter = Counter + 1;   // increment Counter by 1
  delay(500);              // wait for half a second
}
```

These lines are first comparing if the “Button_state” variable has detected a “LOW” (a pressed state). And if this is true, then increment the Counter variable by 1 (which was initially 0) and then wait for half a second (500ms).

```
if(Counter == 1)                // if Counter value is 1
{
  digitalWrite(LED1,HIGH);// turn On first LED
}

if(Counter == 2)                // if Counter value is 2
{
  digitalWrite(LED2,HIGH);// turn On second LED
}

if(Counter == 3)                // if Counter value is 3
{
  digitalWrite(LED3,HIGH);// turn On third LED
}

if(Counter == 4)                // if Counter value is 4
{
  digitalWrite(LED1,LOW);       // turn Off first LED
  digitalWrite(LED2,LOW);       // turn Off second LED
  digitalWrite(LED3,LOW);       // turn Off third LED
  Counter = 0;                  // reset Counter to 0
}
```

The above lines are doing the following:

1. If the current Counter value is 1, turn On the 1st LED.
2. If the current Counter value is 2, turn On the 2nd LED.
3. If the current Counter value is 3, turn On the 3rd LED.
4. If the current Counter value is 4, turn Off all the LEDs.
5. Reset the Counter to 0.

Note that throughout the program, we have used “=” for assigning a value to a variable while “==” has been used for comparison and the comparison is always inside the braces of an “if” statement such as:

```
if(Counter == 3)
```

Here, we are comparing if the value of the variable “Counter” is equal to 3 or not. If it is 3, this “if” gives output as True. If, however, it is not 3, this “if” gives output as False.

```
Counter = 0;
```

Here, we are assigning the variable “Counter” the value 0.

There is no necessity to stop at number 3 or 4. The integer variable “Counter” can very well accommodate values up to 65535. We are, however, limited by:

1. Number of LEDs available.
2. Number of Arduino Pins available to connect the LEDs.

Outcome and Observations:

1. For Part 1, once you are done compiling and uploading the code to the Arduino, you will observe the following:

    a. When you press the button and keep it pressed, the on-board LED turns On and Stays On as long as the button remains pressed.

    b. When you release the button and it stays released, the on-board LED turns Off and Stays Off as long as the button remains released.

    c. No matter how fast or slow you press the button, the LED stays On when the button is pressed and stays Off when the button is released.

    This type of behavior is called “Momentary” because the state of the output (the LED) entirely depends on the state of the Input (the push button). Button On makes LED On and Button Off makes LED Off.

2. For Part 2, once you are done compiling and uploading the code to the Arduino, you will observe the following:

    a. Initially, all the 3 LEDs remain Off.

    b. When the push button is pressed for the first time, the Counter value increments from 0 to 1 and thus turns the 1st LED On.

    c. When the push button is pressed for the second time, the Counter value increments from 1 to 2 and thus turns the 2nd LED On.

    d. When the push button is pressed for the third time, the Counter value increments from 2 to 3 and thus turns the 3rd LED On.

    e. When the push button is pressed for the fourth time, the Counter value increments from 3 to 4 and thus turns all the LEDs Off. Additionally, the Counter value is also re-initialized to 0. This operation readies the entire setup to start over again.