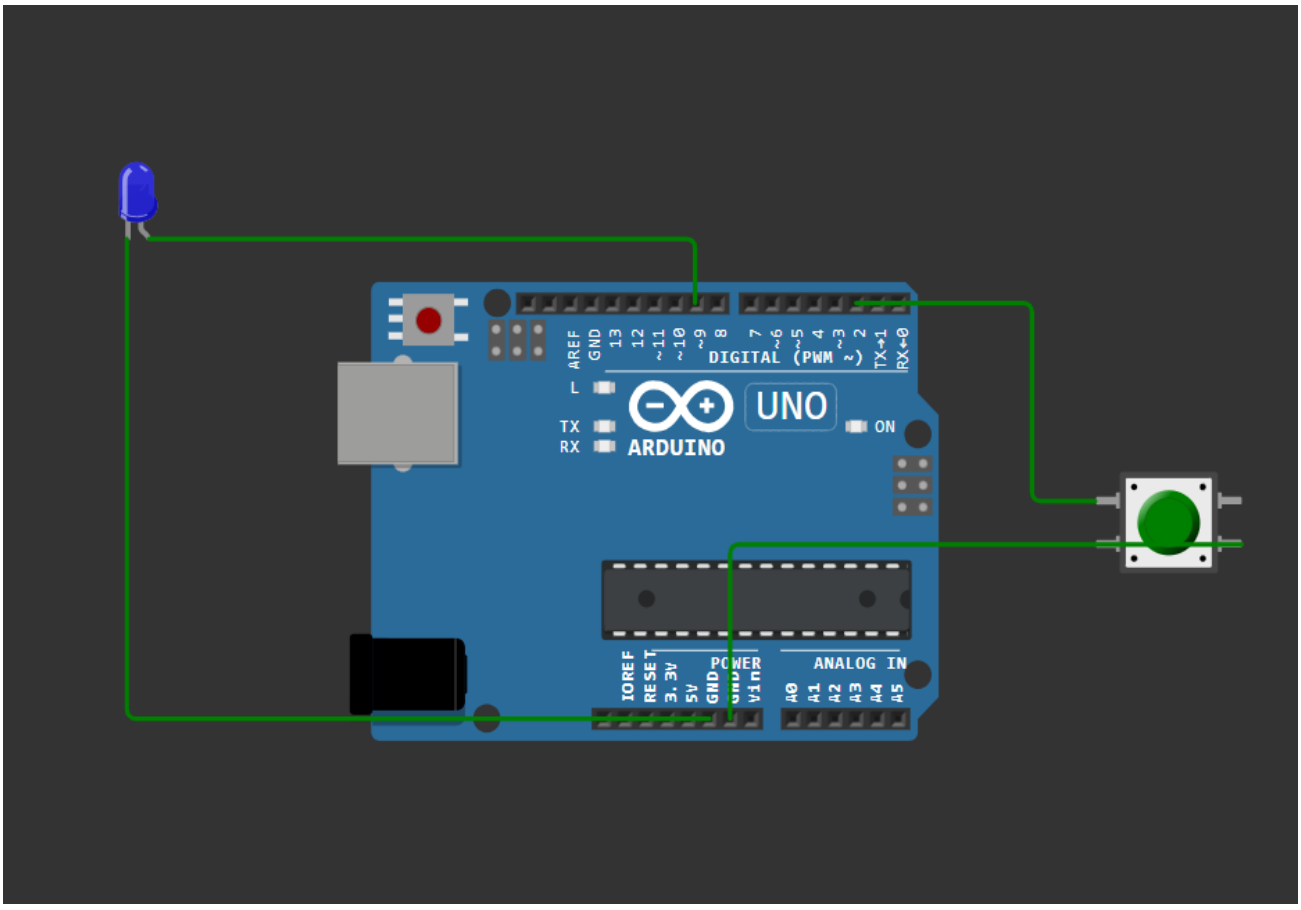
Connecting a button

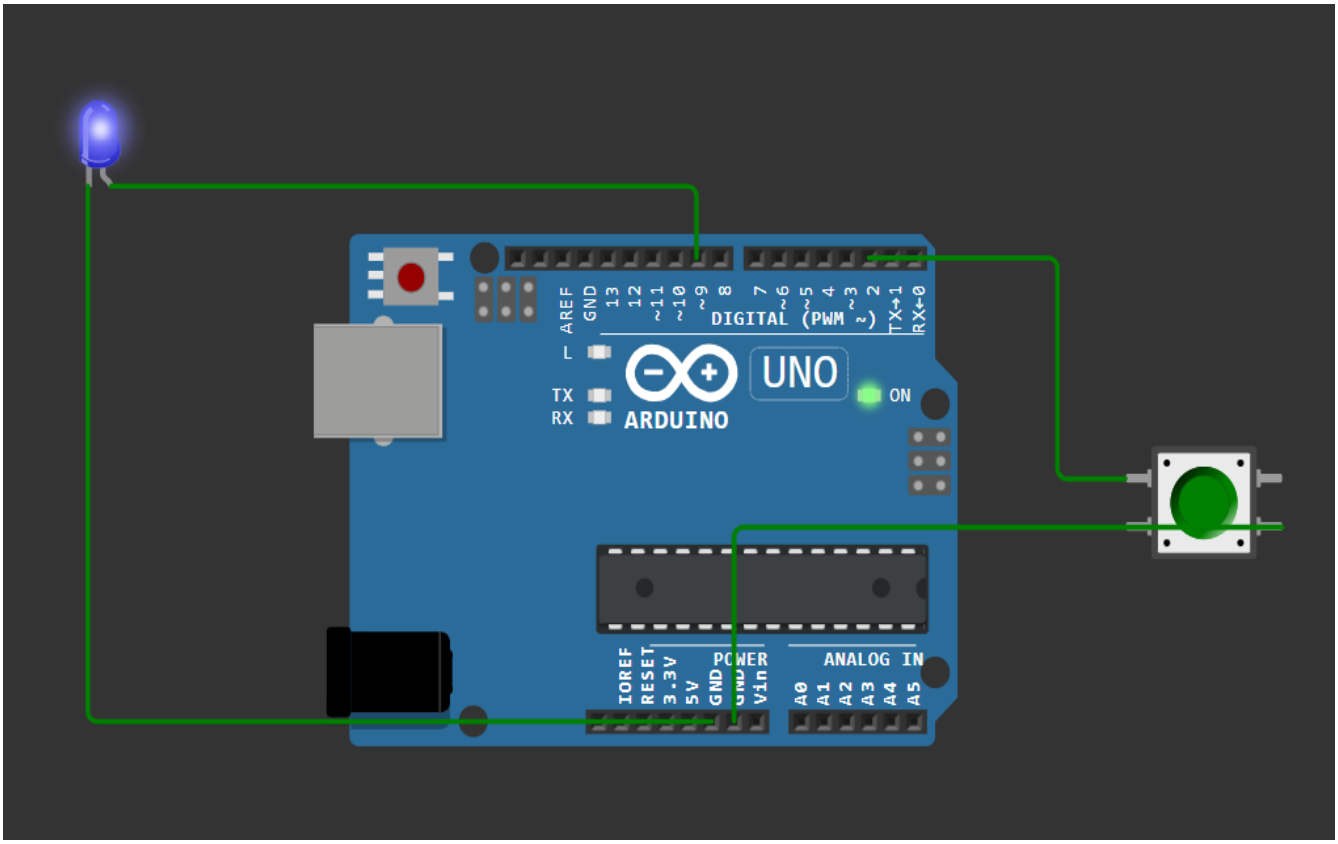
Button needed to be pressed and hold

```
const int buttonPin = 2;
const int ledPin = 9;

int buttonState = 0;
bool ledState = false;

unsigned long lastDebounceTime = 0;
unsigned long debounceDelay = 50;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);
}

void loop() {
  int reading = digitalRead(buttonPin);
  if (reading == LOW) {
    reading = HIGH;
  } else {
    reading = LOW;
  }
  if (reading != buttonState) {
    lastDebounceTime = millis();
  }

  if ((millis() - lastDebounceTime) > debounceDelay) {
    if (reading != ledState) {
```

```
      ledState = reading;
      digitalWrite(ledPin, ledState ? HIGH : LOW);
    }
  }

  buttonState = reading;
}
```



Button is not pressed, light is off

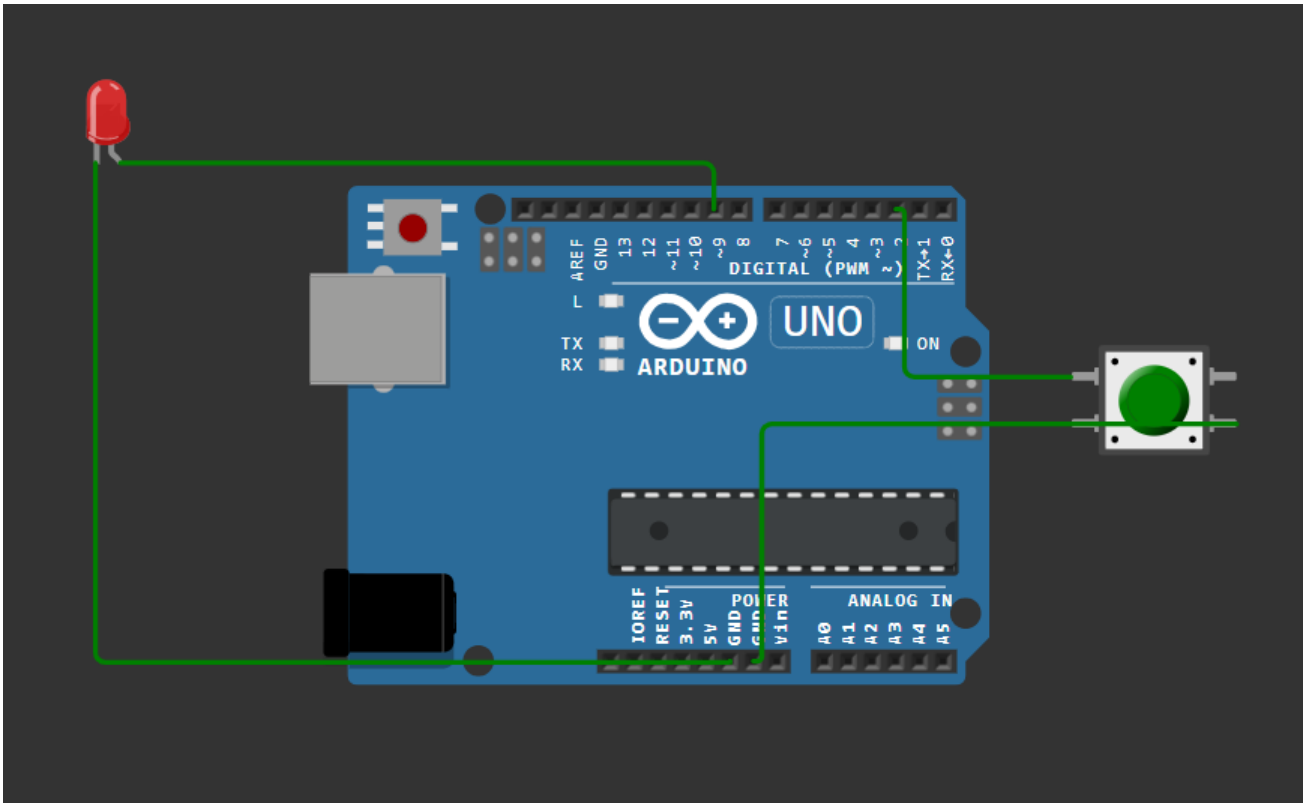


Button is pressed and hold, light is on

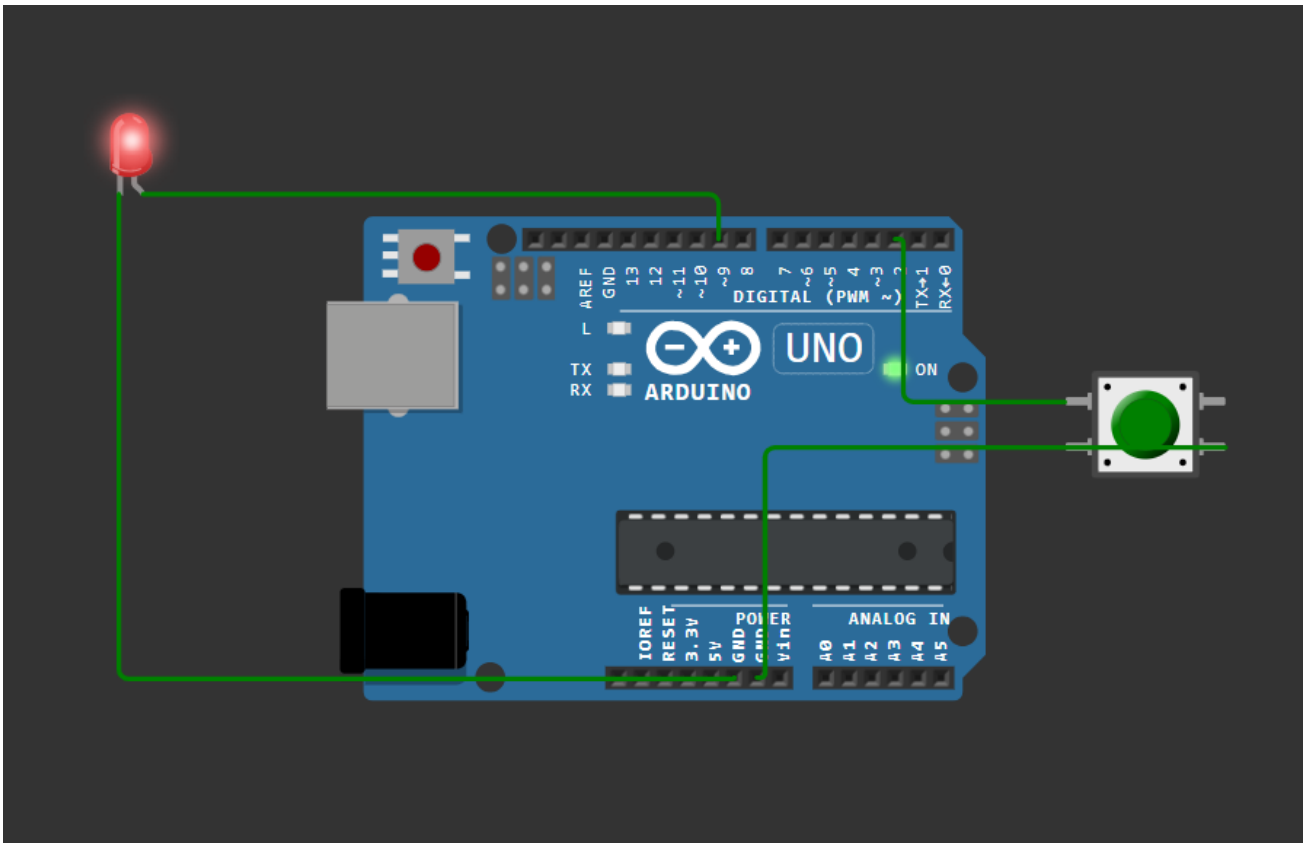
Button only needs one push

```
const int buttonPin = 2;

const int ledPin = 9;


int buttonState;

int lastButtonState = LOW;

bool ledState = false;
```

```
unsigned long lastDebounceTime = 0;
unsigned long debounceDelay = 50;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);
}

void loop() {
  int reading = digitalRead(buttonPin);
  if (reading != lastButtonState) {
    lastDebounceTime = millis();
  }
  if ((millis() - lastDebounceTime) > debounceDelay) {
    if (reading == LOW && buttonState == HIGH) {
      ledState = !ledState;
      digitalWrite(ledPin, ledState ? HIGH : LOW);
    }
    buttonState = reading;
  }

  lastButtonState = reading;
}
```



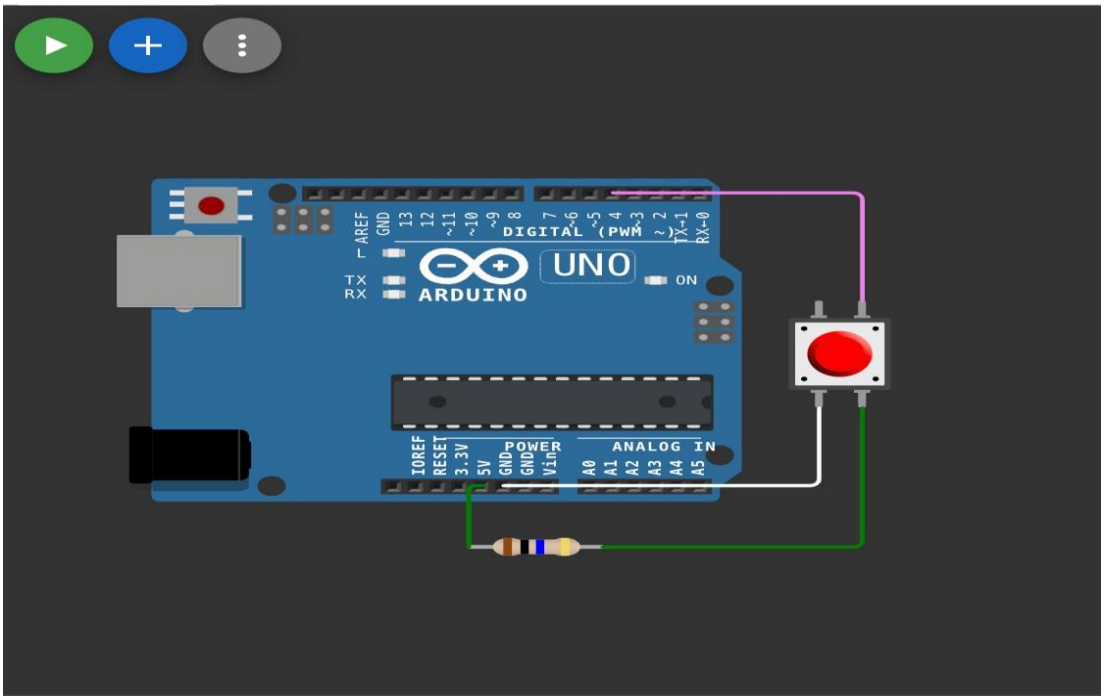
Light is off before push



Light is on after push

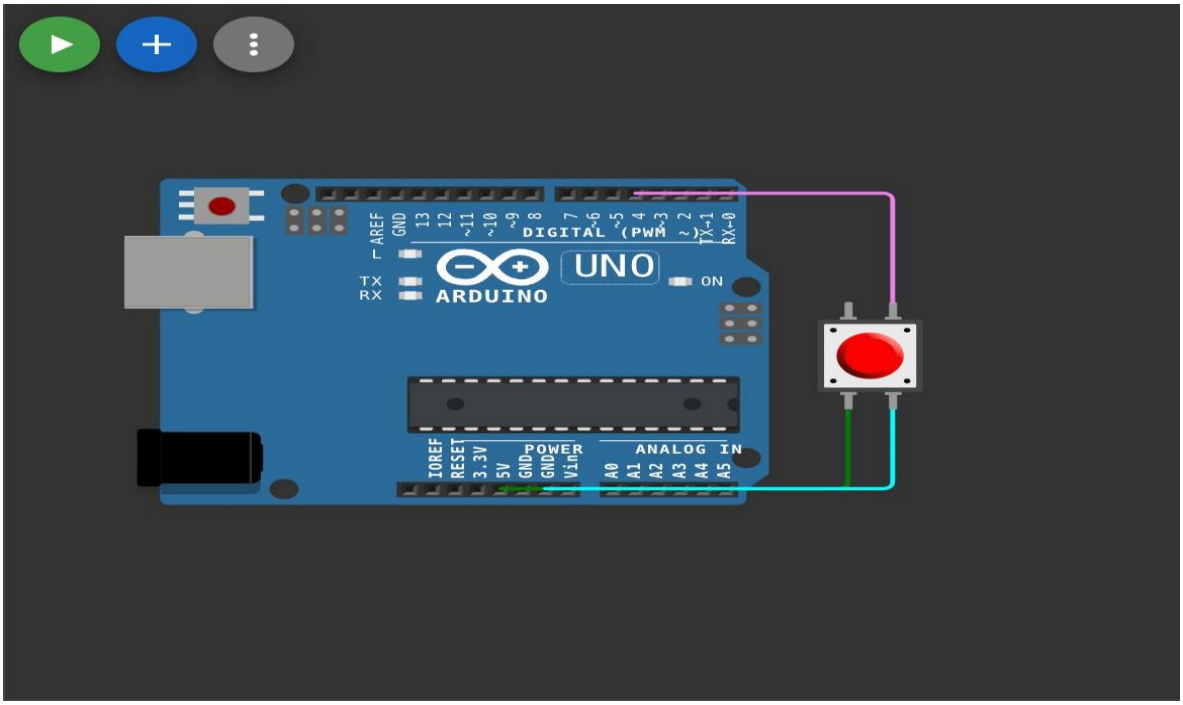
Button with resistor

```
#define BUTTON_PIN 4

void setup()

{

  Serial.begin(9600);

  pinMode(BUTTON_PIN, INPUT_PULLUP);

}

void loop()

{

  byte buttonState = digitalRead(BUTTON_PIN);


  if (buttonState == LOW) {

     Serial.println("Button is pressed");

  }

  else {

     Serial.println("Button is not pressed");

  }

  delay(1000);

}
```



Button with Resistor

```
#define BUTTON_PIN 4

void setup()

{

  Serial.begin(9600);

  pinMode(BUTTON_PIN, INPUT_PULLUP);

}

void loop()

{

  byte buttonState = digitalRead(BUTTON_PIN);


  if (buttonState == HIGH) {

     Serial.println("Button is pressed");

  }

  else {

     Serial.println("Button is not pressed");

  }

  delay(1000);

}
```



Bounce Part

```
#define BUTTON_PIN 4


byte lastButtonState = LOW;


void setup() {
  Serial.begin(9600);
  pinMode(BUTTON_PIN, INPUT);
}


void loop() {
  byte buttonState = digitalRead(BUTTON_PIN);
  if (buttonState != lastButtonState) {
    lastButtonState = buttonState;
    if (buttonState == LOW) {
      Serial.println("Button released");
    }
  }
}
```

Debounce Part

```
#define BUTTON_PIN 4


byte lastButtonState = LOW;


unsigned long debounceDuration = 50; // millis
unsigned long lastTimeButtonStateChanged = 0;


void setup() {
  Serial.begin(9600);
  pinMode(BUTTON_PIN, INPUT);
}


void loop() {
  if (millis() - lastTimeButtonStateChanged > debounceDuration) {
      byte buttonState = digitalRead(BUTTON_PIN);
      if (buttonState != lastButtonState) {
      lastTimeButtonStateChanged = millis();
      lastButtonState = buttonState;
      if (buttonState == LOW) {
        Serial.println("Button released");
      }
    }
  }
}
```

Some Important Terms :

1. Bounce

- **Definition:** Bounce is a physical effect where a push button or switch creates rapid, unintended on/off signals when pressed or released. This occurs because the metal contacts inside the button don't connect cleanly and may vibrate momentarily, causing multiple signals.
- **Impact:** Without handling bounce, a button press could register multiple presses, leading to erratic behavior in electronic circuits.

2. Debounce

- **Definition:** Debounce is the process of eliminating or smoothing out the rapid, unintended signals caused by bouncing in a button or switch. In software, this is typically handled by adding a small delay or by detecting only stable button states.
- **Purpose:** Debouncing ensures that a single, clean signal is registered for each press or release, allowing more accurate button readings.

3. Pull-Up Resistor

- **Definition:** A Pull-Up Resistor is a resistor connected between an input pin and the positive supply voltage (e.g., 5V) to keep the pin at a HIGH voltage level when the button or switch is open (not pressed). When the button is pressed, it connects the pin to ground (0V), pulling the input to LOW.
- **Usage:** Pull-up resistors are commonly used with microcontroller input pins to define a known voltage state when a button is not pressed.

4. Pull-Down Resistor

- **Definition:** A Pull-Down Resistor is a resistor connected between an input pin and ground (0V) to keep the pin at a LOW voltage level when the button or switch is open. When the button is pressed, it connects the pin to the positive supply voltage, pulling the input to HIGH.
- **Usage:** Pull-down resistors are used to ensure an input pin reads a LOW state when a button is not pressed, providing a stable reference voltage.