Title of Experiment:

Introduction to Arduino and Digital I/O Control (Single LED & Push Button)

Aim:

To study digital input/output operation using Arduino by interfacing a single LED and push button, and to observe the response on the Serial Monitor.

Apparatus:

- Arduino Uno board
- USB cable
- Breadboard
- 1 LED
- 1 Push button
- 1 Resistor – 220Ω (for LED)
- 1 Resistor – 10kΩ (for button pull-down)
- Jumper wires
- Arduino IDE installed on PC

Precautions:

1. Connect components with the correct polarity.
2. Use a resistor with the LED to avoid burning it out.
3. Avoid short circuits by checking connections before powering the circuit.
4. Do not upload the program when the circuit is unstable or has loose connections.
5. Disconnect the USB cable before rewiring the setup.

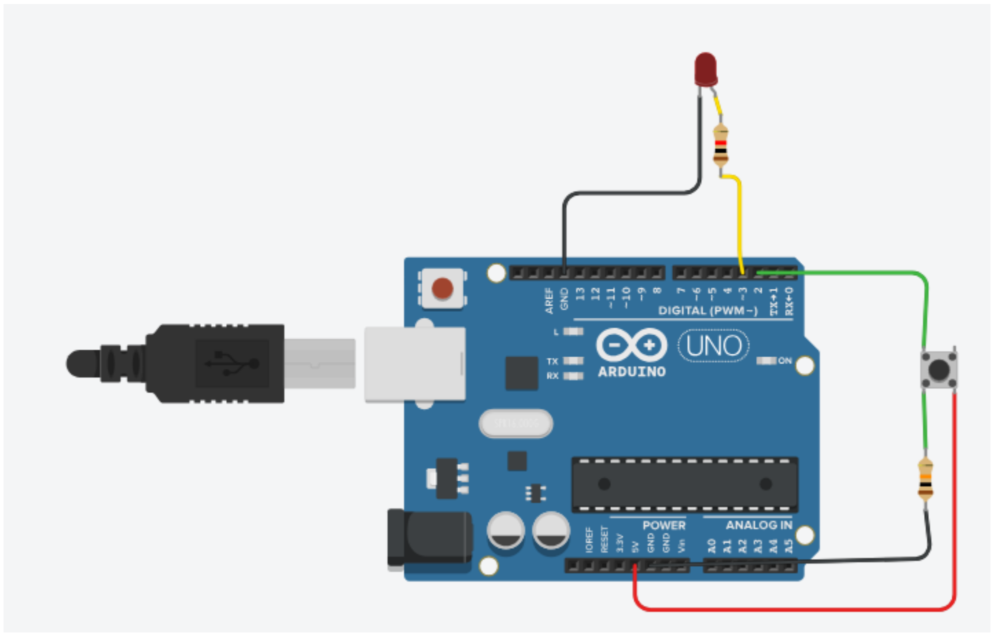
Theory:

Arduino's **digital I/O** capability allows it to interact with the external world.

- **Digital Input**: Reads HIGH (1) or LOW (0) signal, like from a push button.
- **Digital Output**: Sends HIGH or LOW signal, like to turn an LED ON or OFF.

- A **pull-down resistor** ensures a defined LOW state when the button is not pressed.
- The **Serial Monitor** helps observe real-time input/output behavior.

Circuit Diagram:



Procedure:

1. Connect the push button between 5V and pin D2 with a 10kΩ resistor from D2 to GND.
2. Connect the LED anode to pin D3 through a 220Ω resistor; cathode to GND.
3. Plug in Arduino via USB and open Arduino IDE.
4. Write the program to turn ON the LED when the button is pressed.
5. Upload the code to the Arduino board.
6. Open Serial Monitor to observe button status.
7. Press and release the button and observe the LED and Serial Monitor.

Arduino Programming

```
// Define pin numbers

const int buttonPin = 2;  // Push button connected to digital pin 2

const int ledPin = 3;     // LED connected to digital pin 3
```

```
void setup() {
  pinMode(buttonPin, INPUT);   // Set pin 2 as INPUT to read button state
  pinMode(ledPin, OUTPUT);     // Set pin 3 as OUTPUT to control the LED
  Serial.begin(9600);          // Start Serial Monitor communication at 9600 baud
}

void loop() {
  int buttonState = digitalRead(buttonPin);  // Read the current state of the button

  if (buttonState == HIGH) {                 // If button is pressed (input is HIGH)
    digitalWrite(ledPin, HIGH);              // Turn the LED ON
    Serial.println("Button Pressed - LED ON");  // Print message to Serial Monitor
  } else {
    digitalWrite(ledPin, LOW);               // Turn the LED OFF if button is not pressed
    Serial.println("Button Released - LED OFF"); // Print message to Serial Monitor
  }

  delay(100);  // Wait for 100 milliseconds to reduce noise (debounce effect)
}
```

Conclusion:

The experiment successfully demonstrated basic digital input and output control using Arduino. The LED responded to the push button press, and the system status was verified through the Serial Monitor. This forms a foundation for understanding user-interface control in embedded systems.

Q1: What is the function of `pinMode()` in Arduino programming?

Answer:
The `pinMode()` function is used to configure a specific pin as either **INPUT** or **OUTPUT**. In this experiment, it sets the push button pin as an input and the LED pin as an output.

Q2: Why do we use a pull-down resistor with the push button?

Answer:
A pull-down resistor (e.g., 10kΩ) ensures that the input pin reads **LOW (0)** when the button is not pressed. Without it, the pin might float and give **unpredictable or noisy readings**.

Q3: What is the purpose of `Serial.begin(9600)` and `Serial.println()` in the code?

Answer:
`Serial.begin(9600)` initializes serial communication between the Arduino and the computer at 9600 baud rate.
`Serial.println()` is used to **display messages** in the Serial Monitor, helping in **debugging and real-time monitoring** of button states.

Q4: What happens in the code when the button is pressed?

Answer:
When the button is pressed, the input pin reads **HIGH**. The program then executes `digitalWrite(ledPin, HIGH);` to **turn the LED ON**, and a message is printed to the Serial Monitor saying **"Button Pressed - LED ON"**.