

# **DIGITAL ASSIGNMENT 1**

**BCSE311P**

**SENSORS AND ACTUATORS**

**G2+TG2**

**VL2024250102120**

*submitted by*

**1. BHARATH A. S. - 21BCT0032**

**2. DESISH D. - 21BCE2299**

**3. SURAJ JHA - 21BCT0358**

**4. ARTH P SINGH- 21BCT024**

**5. KAVI SHAH - 21BCE2083**



**VIT<sup>®</sup>**

**Vellore Institute of Technology**

(Deemed to be University under section 3 of UGC Act, 1956)

## **TABLE OF CONTENTS**

<b>Sl. No.</b>	<b>COMPONENT</b>	<b>PAGE NO.</b>
<b>1.</b>	<b>AIM</b>	<b>3</b>
<b>2.</b>	<b>SCENARIO EXPLANATION</b>	<b>3</b>
<b>3.</b>	<b>COMPONENTS EXPLANATION</b>	<b>3-4</b>
<b>4.</b>	<b>CIRCUIT DIAGRAM AND EXPLANATION</b>	<b>5-6</b>
<b>5.</b>	<b>HARDWARE PROTOTYPE PHOTOS</b>	<b>6-8</b>
<b>6.</b>	<b>ARDUINO CODE</b>	<b>9=10</b>
<b>7.</b>	<b>PSEUDOCODE/FLOWCHART</b>	<b>10-11</b>
<b>8.</b>	<b>INFERENCE</b>	<b>12</b>

## 1. AIM

To Implement a push-button activated LED light-up circuit using Arduino uno microcontroller (wherein the interfaced LED lights up only when the push-button, interfaced via an external breadboard, is pressed).

## 2. SCENARIO EXPLANATION

In this project, we'll create a simple circuit that lights up an LED when a push button is pressed. The Arduino microcontroller will be used to read the state of the button and control the LED. This basic project introduces the concept of digital input (button) and output (LED) using an Arduino, helping to understand how microcontrollers interact with external components.

When the button is pressed, it completes the circuit, sending a signal to the Arduino, which then turns on the LED. Releasing the button breaks the circuit, turning the LED off. This project demonstrates how to use digital pins on the Arduino to read inputs and control outputs, which is fundamental in building more complex systems.

## 3. COMPONENT EXPLANATION

To build this project, the following components were used:

### - Arduino Uno:

#### **Microcontroller Used in Arduino LED Interfacing**

ATmega328P Microcontroller

The microcontroller at the heart of the Arduino Uno is the ATmega328P . This is an 8-bit microcontroller from the AVR family, developed by Microchip Technology. It is widely used in embedded systems and hobby electronics due to its versatility and ease of use. Here are some key features of the ATmega328P:

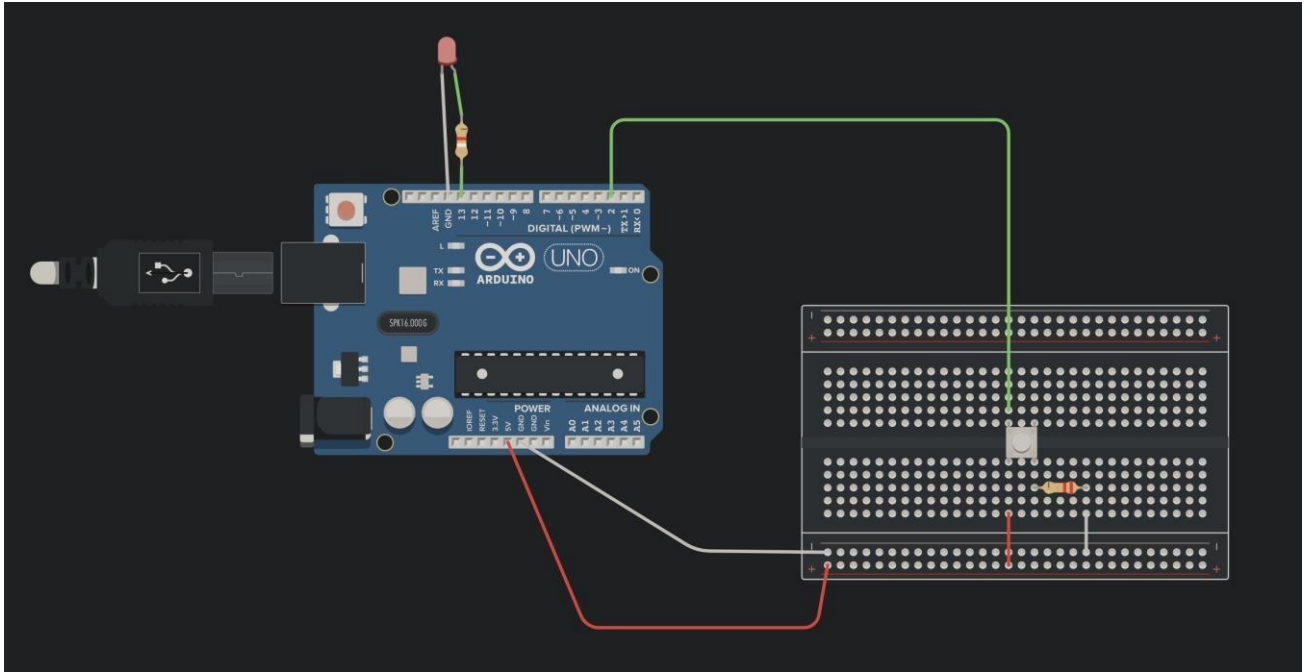
- Architecture : 8-bit RISC (Reduced Instruction Set Computing), which offers efficient use of instruction cycles.
- Operating Voltage : 1.8V to 5.5V, with 5V being the typical operating voltage in Arduino Uno.

- Digital I/O Pins : 14 pins that can be used as input or output pins, allowing the control of devices like LEDs and reading inputs from sensors.
- Analog Input Pins : 6 analog pins for reading signals from analog sensors.
- Flash Memory : 32 KB for storing the program code.
- SRAM : 2 KB for storing variables during program execution.
- EEPROM : 1 KB for non-volatile storage of data that must persist across resets.
- Clock Speed : 16 MHz, which provides a good balance between performance and power consumption.
- Timers and PWM : Three timers (two 8-bit and one 16-bit) and 6 PWM (Pulse Width Modulation) outputs, useful for controlling motors, LEDs, etc.
- Serial Communication : Supports UART, SPI, and I2C communication protocols.

The ATmega328P's simplicity and functionality make it ideal for beginners learning about microcontrollers and for projects that require a reliable microcontroller to control various inputs and outputs.

- LED: A light-emitting diode that will light up when the button is pressed. LEDs have two leads; the longer lead is the anode (+), and the shorter one is the cathode (-).
- Push Button: A momentary switch used to control the flow of electricity in the circuit. When pressed, it completes the circuit, sending a signal to the Arduino.
- Resistors:
  - 220-ohm resistor: Used to limit the current flowing through the LED to prevent it from burning out.
  - 10k-ohm resistor: Used as a pull-down resistor for the push button to ensure it reads a definite low signal when not pressed.
- Breadboard: A tool for making temporary circuits and prototyping without soldering.
- Jumper Wires: Used to connect components on the breadboard and to the Arduino microcontroller.

#### 4. CIRCUIT DIAGRAM AND EXPLANATION



The circuit setup involves connecting the components to the Arduino as follows:

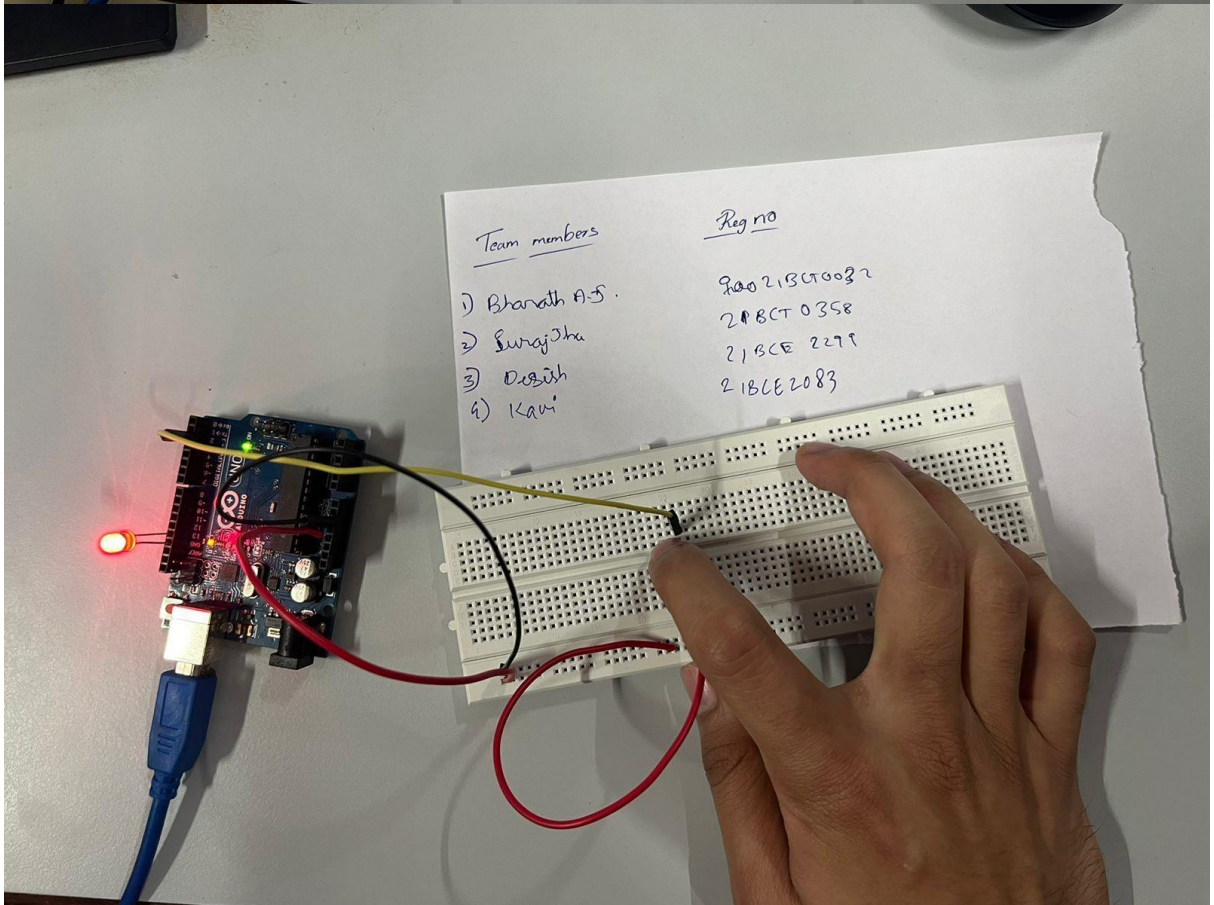
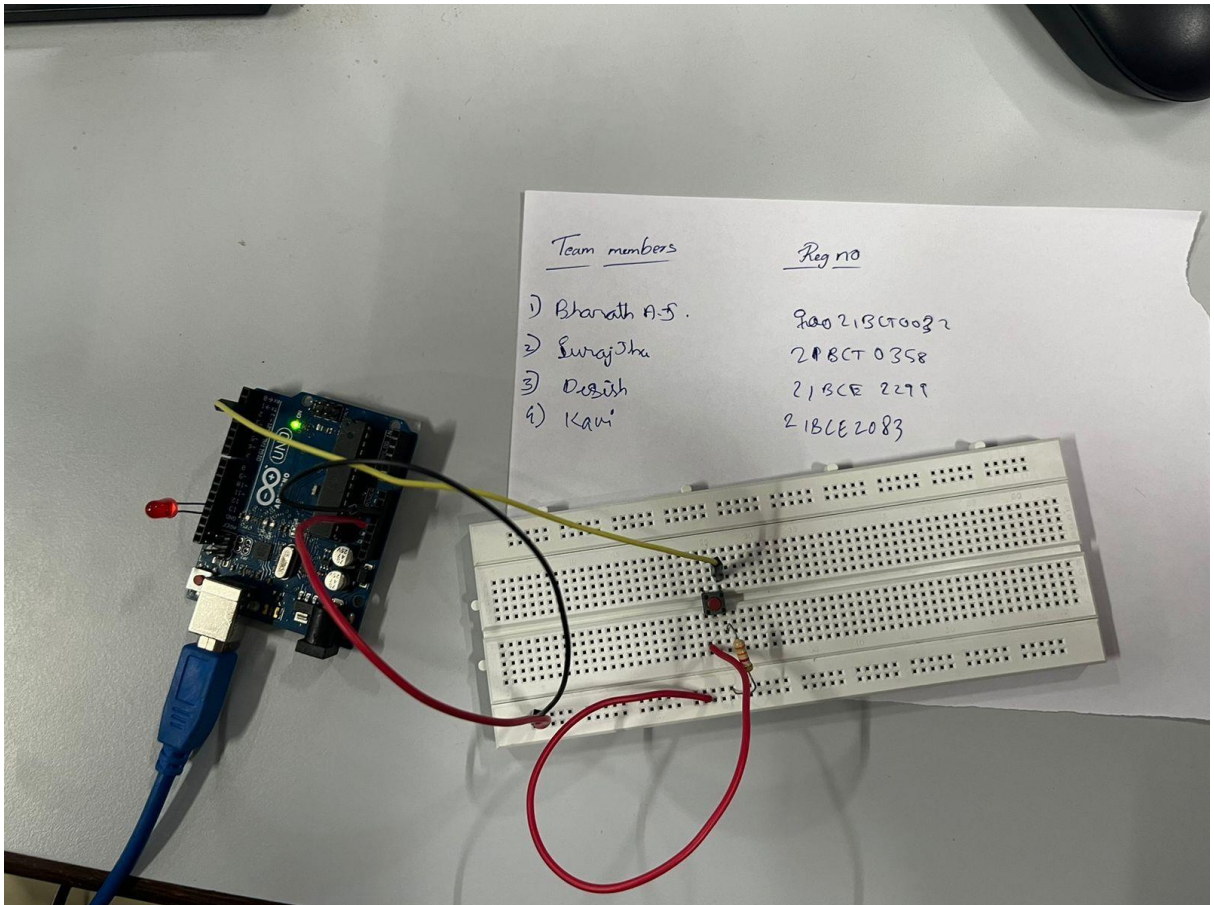
- **Connecting the Breadboard and Arduino Board**
  - Voltage Line:  
Connect the Voltage rail (+) of the Breadboard to the 3.3V supply of the Arduino Power Module using a jumper wire (red wire in the image).
  - Ground Line: Connect the Ground rail (-) of the Breadboard to one of the 2 Ground pins of the Arduino Power Module using a jumper wire (large white wire in the image).

## Connecting the Push-button

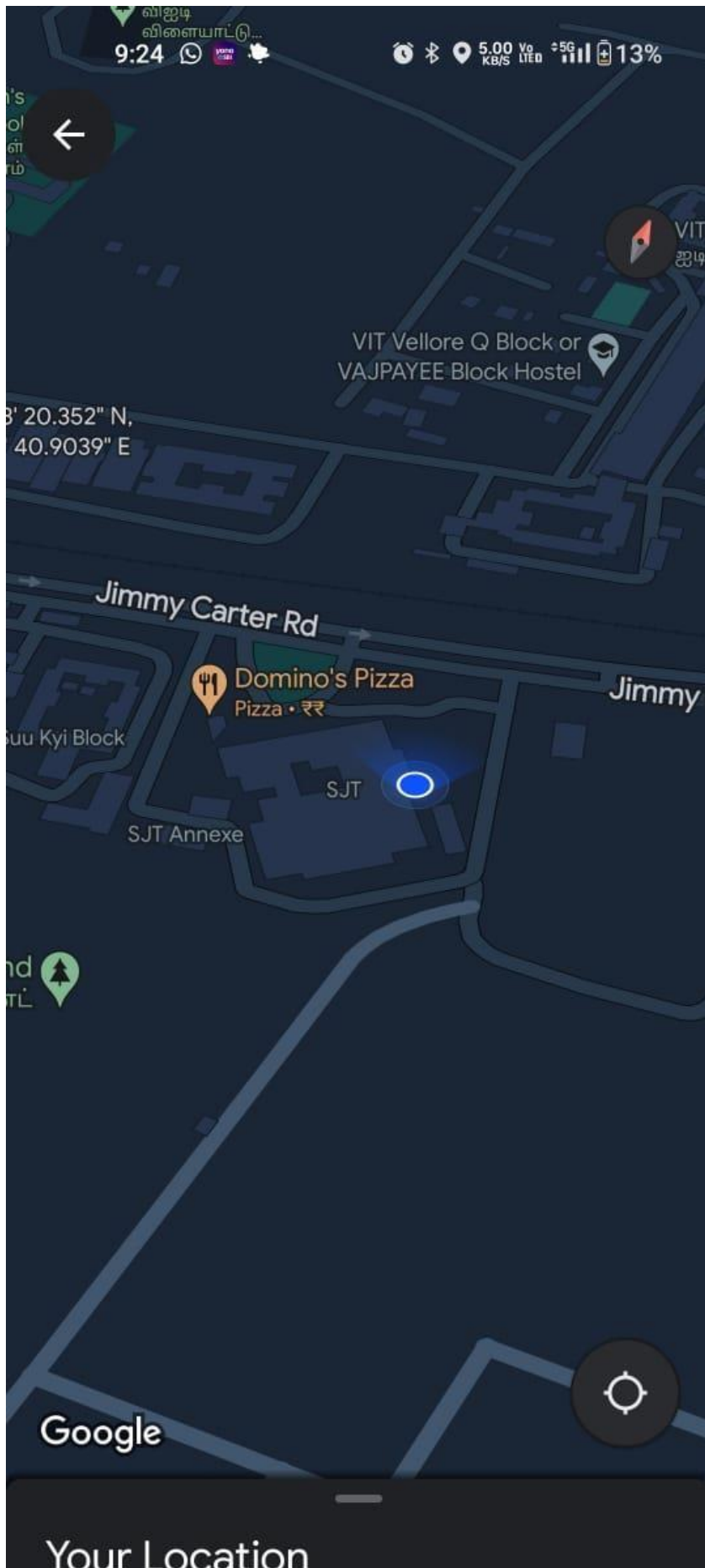
- ○ Push-button Placement: Place the push-button on the breadboard. Ensure the push-button's legs straddle the central divide of the breadboard.
  - Connecting the Button to Arduino Pin: Connect one leg of the pushbutton to digital pin 2 on the Arduino using a jumper wire (green wire in the image).
  - Grounding the Button: Connect the other leg of the push-button to ground (GND) using a jumper wire (white wire in the image). Add a  $10k\Omega$  resistor from the same leg of the push-button to the ground. This serves as a pull-down resistor to ensure the button reads LOW when not pressed.
- **Connecting the LED** ○ LED Placement: Insert the LED onto the breadboard. Ensure that the longer leg (anode, +) is connected to the correct circuit path and the shorter leg (cathode, -) is connected to ground.
- - Connecting the LED to Arduino Pin: Connect the anode (+) of the LED to digital pin 13 on the Arduino using a jumper wire (black wire in the image).
  - Connecting the LED to Ground: Connect the cathode (-) of the LED to the ground (GND) using a jumper wire (red wire in the image).

This configuration ensures that when the button is pressed, pin 2 receives a high signal, turning on the LED connected to pin 13.

## 5. HARDWARE PROTOTYPE

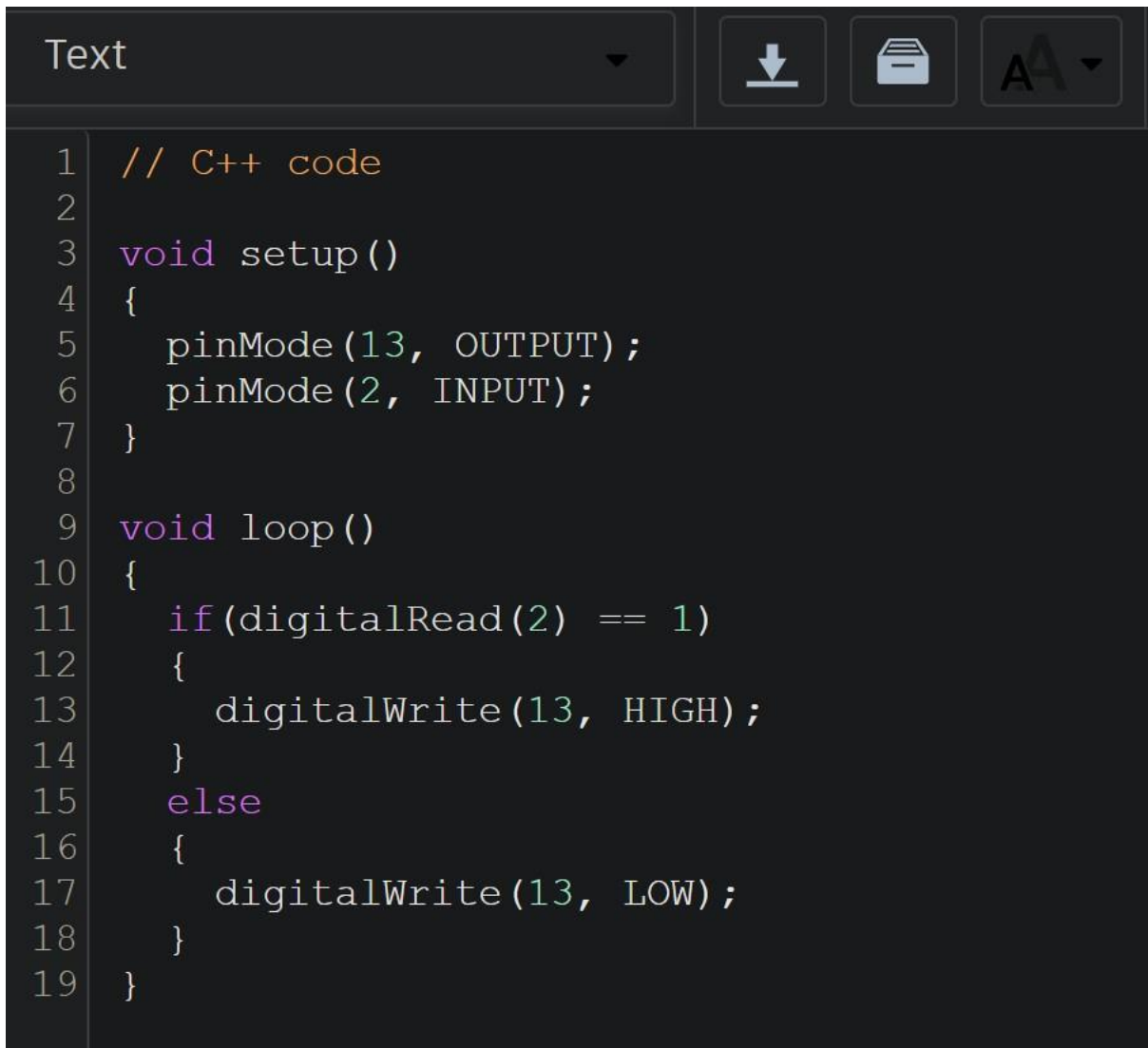








## 6. ARDUINO CODE

A screenshot of an Arduino IDE code editor. The interface has a dark theme. At the top, there is a toolbar with a dropdown menu labeled 'Text', a download icon, a save icon, and a font size selector showing 'A4'. The code is written in a monospaced font and is color-coded: comments are orange, keywords are purple, and string literals are green. The code is as follows:

```
1  // C++ code
2
3  void setup()
4  {
5      pinMode(13, OUTPUT);
6      pinMode(2, INPUT);
7  }
8
9  void loop()
10 {
11     if(digitalRead(2) == 1)
12     {
13         digitalWrite(13, HIGH);
14     }
15     else
16     {
17         digitalWrite(13, LOW);
18     }
19 }
```

```
// C++ code
```

```
void setup()
```

```
{
```

```
    pinMode(13, OUTPUT);
```

```
    pinMode(2, INPUT);
```

```
}
```

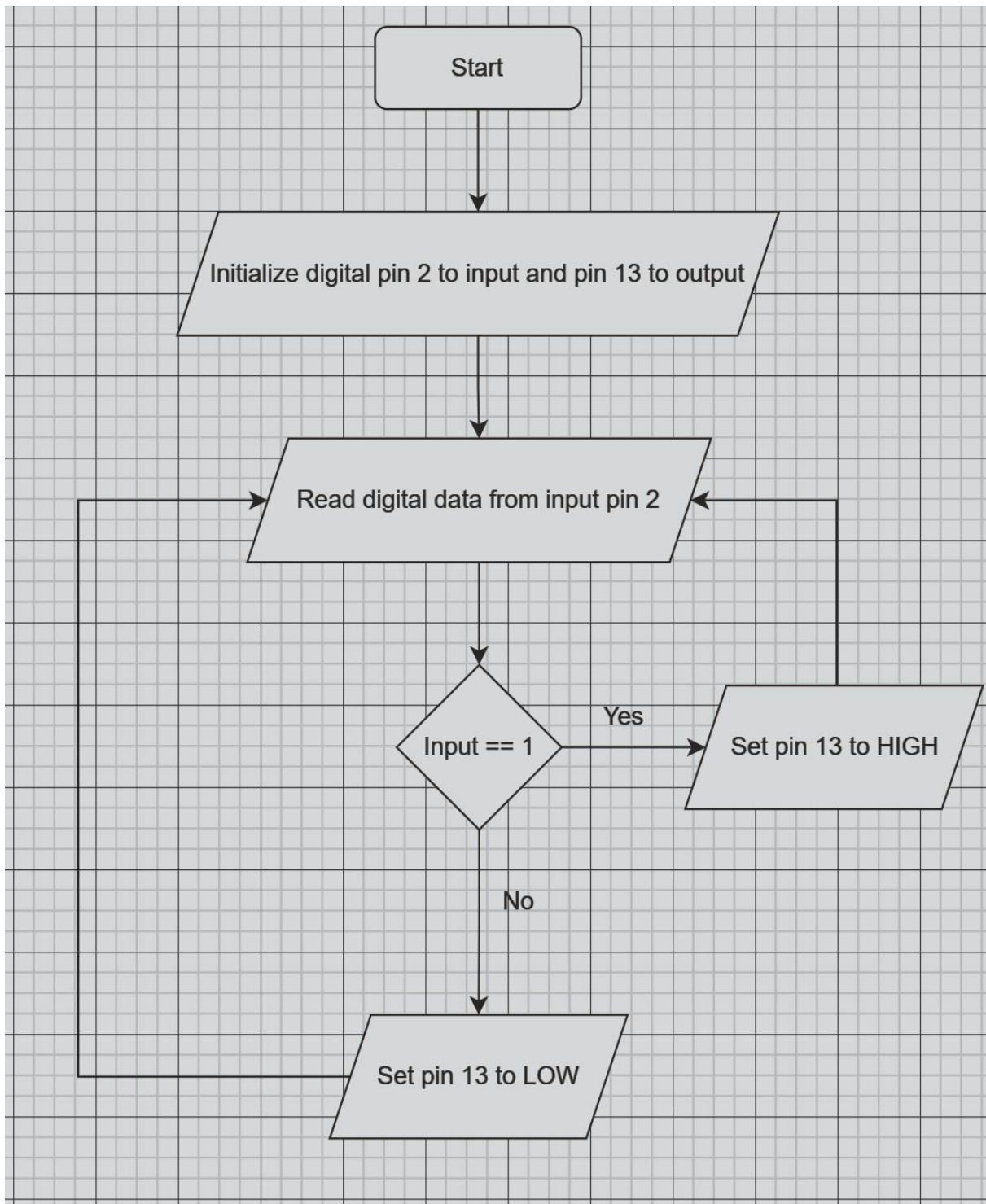
```

void loop() {
  if(digitalRead(2) == 1)
  {
    digitalWrite(13, HIGH);
  }
  else
  {
    digitalWrite(13, LOW);
  }
}

```

## **7. PSEUDOCODE/FLOWCHART**

1. Start
2. Setup:
  - Initialize LED pin as OUTPUT.
  - Initialize button pin as INPUT.
3. Loop:
  - Read the button state.
  - If the button is pressed (HIGH):
    - Turn on the LED.
  - Else:
    - Turn off the LED.
4. Repeat Loop



This flowchart represents the basic logic of the program, where the Arduino continuously checks the button state and controls the LED accordingly.

## **8. INFERENCE**

Thus Implemented a push-button activated LED light-up circuit, using Arduino UNO Microcontroller via a breadboard.