**Practical 4**

**Aim:- Understand the working principle of a 7-segment display and learn how to control and display numerical values**

**Theory**

**7-Segment Display (SSD)**

A 7-segment display is an electronic display device used to display decimal numbers and some letters. It consists of 7 LEDs arranged in a figure-eight pattern and can be controlled to display numbers 0-9 by turning on specific segments.

**Arduino**

The Arduino is an open-source microcontroller platform used to control various electronic components. It can be programmed to read inputs, control outputs, and interact with other hardware.

**3. Materials Required**

* Arduino Uno board
* 7-Segment Display (Common Cathode)
* 220-ohm resistors (for LED and SSD)
* Breadboard
* Jumper wires

**Procedure**

**Connecting the 7-Segment Display**

1. **Identify Pins:**
   * The 7-segment display typically has 8 pins: 7 for each segment (labeled A to G) and one common cathode pin.
2. **Connect the 7-Segment Display:**
   * Connect the segment pins (A to G) to Arduino digital pins (e.g., 2 to 8).
   * Connect the common cathode pin to GND through a 220-ohm resistor.

**Code**

**Arduino Sketch for 7 Segment display**

**#include "SevSeg.h"**

**SevSeg sevseg;**

**const int  buttonPin = 10;    // the pin that the pushbutton is attached to**

**int buttonState = 0;          // current state of the button**

**int lastButtonState = LOW;    // previous state of the button**

**int buttonPushCounter = 0;    // counter for the number of button presses**

**long counter = 0;**

**long max\_long\_val = 2147483647L;**

**void setup(){**

**byte numDigits = 1;**

**byte digitPins[] = {};**

**byte segmentPins[] = {6, 5, 2 , 3, 4 , 7, 8, 9};**

**bool resistorsOnSegments = true;**

**byte hardwareConfig = COMMON\_CATHODE;**

**sevseg.begin(hardwareConfig, numDigits, digitPins, segmentPins, resistorsOnSegments);**

**sevseg.setBrightness(90);**

**pinMode(buttonPin, INPUT\_PULLUP);**

**Serial.begin(9600);**

**lastButtonState = LOW;**

**}**

**void loop(){**

**buttonState = digitalRead(buttonPin);**

**if(buttonState == HIGH){**

**buttonState = LOW;**

**}**

**else**

**buttonState = HIGH;**

**if(buttonState == HIGH){**

**Serial.println("on");**

**lastButtonState = HIGH;**

**buttonPushCounter++;**

**if(counter < max\_long\_val)**

**counter++;**

**buttonPushCounter %= 9;**

**sevseg.setNumber(buttonPushCounter, 1);**

**sevseg.refreshDisplay();**

**delay(100 - (counter%99));**

**}**

**else{**

**Serial.println("on");**

**if(lastButtonState == HIGH){**

**Serial.println("in");**

**buttonPushCounter++;**

**buttonPushCounter %= 7;**

**if(buttonPushCounter == 0)**

**buttonPushCounter = 1;**

**counter--;**

**sevseg.setNumber(buttonPushCounter, 1);**

**sevseg.refreshDisplay();**

**delay(100 - (counter%99));**

**if(counter == 0){**

**lastButtonState = LOW;**

**sevseg.setNumber(buttonPushCounter, 0);**

**sevseg.refreshDisplay();**

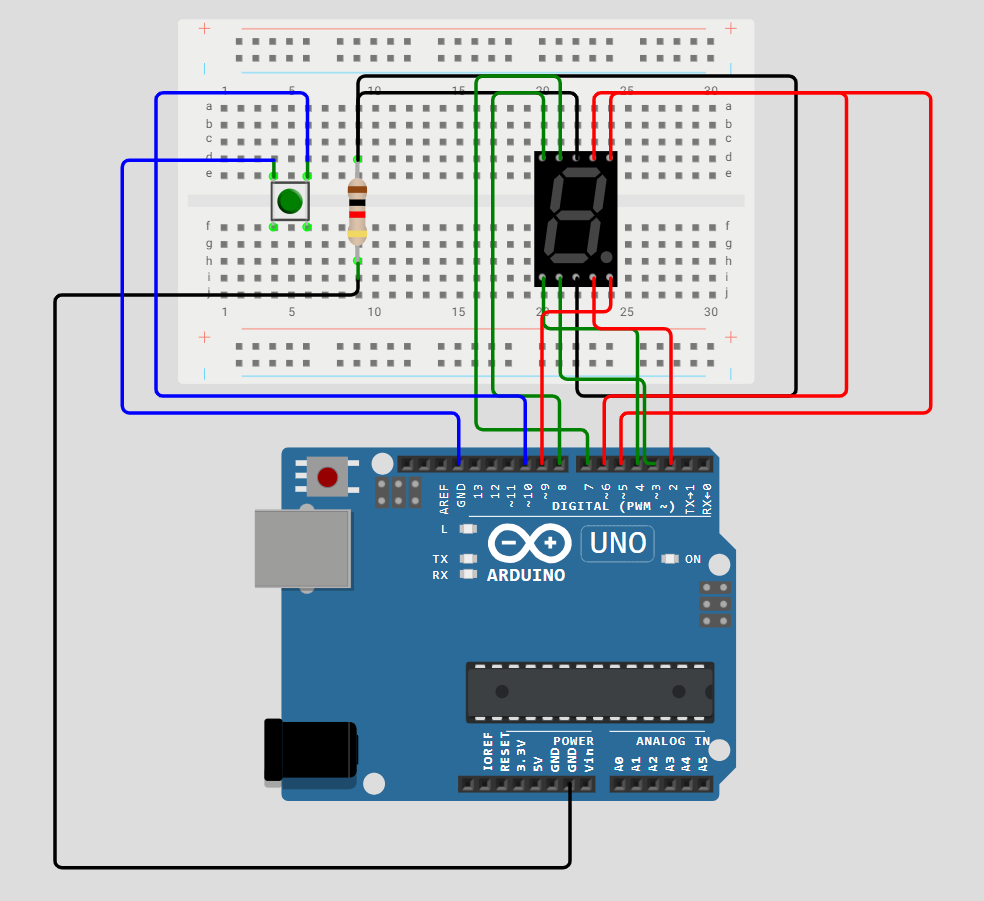
**}**

**}**

**}**

**}**

**Image**



*Fig;- Arduino board with 7-segment display connected.*

**Working**

**7-Segment Display Control:**

The 7-segment display shows digits 0 through 9 sequentially, each for 2 seconds. The displayDigit() function uses predefined patterns to illuminate the appropriate segments of the display to represent each digit.

**Conclusion**

The lab successfully demonstrated how to control an LED and a 7-segment display using an Arduino. Students learned to wire up basic electronic components and write code to control them, understanding the concepts of digital output and display management. This exercise provided foundational skills in interfacing components with an Arduino.