**Project Based Learning Report**

on

**“T****wo Digit Counter Using Arduino & 7 Segment Display & Push Button”**

Submitted in the partial fulfilment of the requirements

For the Project based learning in **Embedded System Design**

in

Electronics & Communication Engineering

By

**Name of the student PRN**

Ajay Kumar Gupta 2114110451

MD. Gafrujama Ansari 2114110431

Ram Prakash 2214110681

Under the guidance of Course In-charge

Dr. Arundhati A.Shinde

# Department of Electronics & Communication

**Engineering**

Bharati Vidyapeeth

(Deemed to be University)

College of Engineering,

Pune – 411043

Year 2023-24

## Bharati Vidyapeeth (Deemed to be University)

## College of Engineering,

**Pune – 411043**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

## CERTIFICATE

Certified that the Project Based Learning report entitled, “Two Digit Counter Using Arduino & 7 Segment Display & Push Button” is a bonafied work done by

|  |  |
| --- | --- |
| **Name of the student** | **PRN** |

**Ajay Kumar Gupta 2114110451**

**Md Gafrujama Ansari 2114110431**

**Ram Prakash 2214110681**

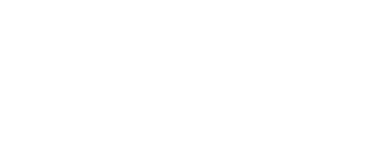
in partial fulfillment of the requirements for the award of credits for Project Based Learning (PBL) in Embedded System Design Course of Bachelor of Technology Semester V, in Electronics and Communication Branch.

Date: **16/10/23**

**Prof. Dr A.A Shinde Prof. Dr. A.A.Shinde Course In-charge Head of department**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Content** | **Page No.** |
| **1.** | Problem statement | **1** |
| **2.** | Project Description | **2-3** |
| **3.** | Software Used | **4-5** |
| **4.** | Applications | **6** |
| **5.** | Algorithm | **7-8** |
| **6.** | Program | **9-10** |
| **7.** | Circuit Diagram | **11** |
| **8.** | Output | **12-13** |
| **9.** | Conclusion & Course Outcome | **14** |
| **10.** | References | **15** |



# PROBLEM STATEMENT

Design and develop Two digit counter using push and button 7 -segment display using arduino.

**Solution.**

**Key Objectives and Features:**

A two-digit counter using push button and 7-segment displays with an Arduino is a popular and educational project that helps we learn about digital electronics and microcontroller programming. Here's a brief description of how this project works:

Components Required:

Arduino Uno

Two 7-segment displays

14 Male-Male Jumper Wires

6 Push Button

Breadboard

Description:

Connect the Displays: Start by connecting the two 7-segment displays to the Arduino. We will need to connect each segment (a, b, c, d, e, f, and g) and the common anode (or common cathode) pins of each display to the appropriate Arduino pins.

Add the Push Button: Attach the six push button to the breadboard and connect them to the Arduino. Three button can be used to control each digit of the counter. One button for increment, one for decrement, and one for resetting the counter to zero. We may need resistors as pull-down (or pull-up) to ensure reliable button state detection.

Arduino Code: Write the Arduino code to control the counter. We'll need to define the pin connections for the 7-segment displays and the push button. Wer code will handle incrementing, decrementing, and resetting the counter based on button presses. It will also update the display to show the current count on the 7-segment displays.

Display Logic: The two 7-segment displays can be used to display numbers from 0 to 99. We'll need to implement logic to manage the transition from a single-digit display to a two-digit display when the count is above 9.

Testing and Calibration: Once the code is uploaded to the Arduino, we can test wer project. Pressing the increment button should increase the count, the decrement button should decrease it, and the reset button should set it to zero. Make sure that the displays show the correct numbers for each count.

This project is a great way to understand how to interface push button and 7-segment displays with an Arduino, and it teaches we the basics of coding and control flow in the Arduino IDE.

# PROJECT DESCRIPTION

A two-digit counter using push button and 7-segment displays is a fundamental project that combines digital input, digital output, and display technology. Here's the theory behind this project:

1. Digital Input with Push Button:

- Push button are used as digital input devices. They have two states: pressed and unpressed. When a button is pressed, it makes an electrical connection, allowing current to flow through it. When it's unpressed, the circuit is open, and no current flows.

2. The Microcontroller (Arduino):

- An Arduino board is used as the central processing unit. It's a microcontroller that can read the state of digital pins and execute code based on those inputs.

3. Counting Logic:

- The Arduino keeps track of a numerical count, which is displayed on the 7-segment displays. The count can range from 0 to 99.

- To increment the count, one push button is dedicated to counting up, and when pressed, the Arduino increments the count.

4. Display Technology:

- 7-segment displays are used to visually represent the numerical count. Each display consists of seven segments (a, b, c, d, e, f, g) that can be illuminated to display numbers and some letters. Common cathode or common anode displays can be used, depending on wer setup.

- The Arduino controls which segments are turned on or off to display the appropriate digits (0-9) on each display.

- For two-digit numbers, we need to display both the tens and ones digits, switching between the displays as necessary.

5. Multiplexing:

- To display numbers on two 7-segment displays with fewer pins, multiplexing is often used. Multiplexing involves rapidly cycling through each display (toggling the common cathode/anode connection) while displaying the appropriate segments to create the illusion that both digits are continuously displayed.

6. Debouncing:

- When button are pressed or released, they can create "bouncing" signals due to mechanical properties. These bouncing signals can result in multiple counts or unstable behavior. To mitigate this, debouncing techniques are applied in software to ensure that only one count is registered per button press.

7. Code Logic:

- The Arduino code handles the following logic:

- Detecting button presses and distinguishing between short presses and long presses (for continuous counting).

- Incrementing the count based on button presses.

- Managing the display logic to show the appropriate digits on the 7-segment displays.

- Implementing a reset function for one of the button.

A two-digit counter using only one push button and a 7-segment display with Arduino is a compact project that demonstrates efficient coding and a simple hardware setup. In this configuration, we'll use a single button press to increment the count from 0 to 99, and we'll display the count on the 7-segment display. Here's how it works:

Hardware Setup:

1. Components:

- Arduino Uno or compatible board

- A single 7-segment display (common cathode or common anode)

- Male-Male Jumper Wires

- One Push Button

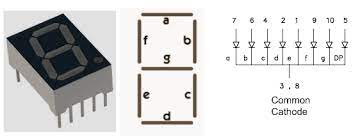
- Breadboard

- Resistors (for button pull-down or pull-up)

2. Connect the Display:

- Connect the cathode (or anode) pins of the 7-segment display to appropriate Arduino digital pins (e.g., D2 to D9).

- Connect the segment pins (a, b, c, d, e, f, and g) to digital pins accordingly.



3. Add the Push Button: - Connect the push button to the breadboard and connect it to an Arduino digital pin. Use a resistor as a pull-up or pull-down, depending on the configuration.

How It Works:

1. The code initializes the pin connections, sets the initial count to 0, and defines the segments for digits 0-9.

2. In the `loop()` function, it continuously checks the state of the push button.

3. When the button is pressed (LOW), it increments the count by 1. The modulo operator ensures that the count resets to 0 after reaching 99.

4. The `updateDisplay()` function is called to update the 7-segment display with the new count.

5. The `updateDisplay()` function displays the ones digit of the count on the 7-segment display.

This project provides a simple way to create a two-digit counter using just one push button and one 7-segment display with an Arduino.

# SOFTWARE USED

Arduino Integrated Development Environment (Arduino IDE) is an open-source software platform used for programming and developing applications for Arduino microcontrollers and compatible boards.



It provides a user-friendly interface and a set of tools that make it easy to write, compile, and upload code to wer Arduino hardware. Here's a description of the key features and components of the Arduino IDE:

1. Code Editor: The primary function of Arduino IDE is to provide a code editor where we can write and edit wer Arduino sketches (programs). It offers features like syntax highlighting and auto-indentation to make coding easier.

2. Compilation: Arduino IDE includes a built-in compiler that translates wer Arduino code (written in C/C++) into machine code that the microcontroller can understand. When we click the "Verify" button, the IDE compiles wer code to check for errors and provide feedback.

3. Uploading: Once we've written and verified wer code, we can upload it to wer Arduino board using the "Upload" button. The IDE handles the compilation process and sends the binary code to the microcontroller via a connected USB cable.

4. Library Manager: Arduino IDE comes with a library manager that allows we to easily include pre-written libraries to extend the functionality of wer Arduino projects. Libraries are collections of code that simplify tasks such as working with sensors, displays, and communication modules.

5. Serial Monitor: Arduino IDE features a built-in serial monitor that allows we to communicate with wer Arduino board and receive data from it. This is especially useful for debugging and monitoring the behavior of wer program.

6. Board Manager: Arduino IDE supports a wide range of Arduino-compatible boards. We can select wer specific board from the "Boards" menu, and the IDE will configure the development environment for that particular board.

7. Integrated Examples: Arduino IDE provides a selection of example sketches for various components and sensors. These examples serve as a great starting point for beginners and can be found under the "File" > "Examples" menu.

8. Multiple Operating Systems: Arduino IDE is compatible with Windows, macOS, and Linux, making it accessible to a broad user base.

9. Community Support: Arduino has a large and active community, and the IDE benefits from a wealth of online resources, forums, and documentation to help users learn and troubleshoot.

10. Open Source: Arduino IDE is open-source software, which means that we can view and modify its source code. This feature allows for customization and expansion of the IDE to suit specific project requirements.

Arduino IDE serves as a valuable tool for both beginners and experienced developers to create a wide range of projects, from simple blinking LEDs to complex IoT applications. It plays a crucial role in simplifying the process of programming Arduino-based microcontrollers and fosters creativity and innovation in the maker and electronics community.

**APPLICATIONS:**

A two-digit counter using push button and 7-segment displays with Arduino has a wide range of practical applications and educational uses. Here are some of the applications where this project can be employed:

1. Educational Tool: This project is an excellent educational tool for teaching digital electronics and microcontroller programming to students. It can be used in classrooms, electronics workshops, and STEM (Science, Technology, Engineering, and Mathematics) programs to introduce concepts like digital input, output, counting, and display technology.

2. Traffic Counter: The project can be adapted for counting vehicles or people at a specific location. For example, it can be used to monitor the number of cars passing through a gate or the number of attendees at an event.

3. Inventory Control: In a warehouse or store, the counter can be used to keep track of the inventory of items. When items are added or removed from a shelf, the counter can be incremented or decremented accordingly.

4. Production Line Counter: In manufacturing, this system can be used to count the number of products produced in a production line. It's especially useful for quality control and production efficiency analysis.

5. Lab Equipment: It can be used in scientific experiments to record the number of occurrences of a particular event. For instance, in a chemistry lab, it can track the number of reactions or measurements taken.

6. Game Scoring: Incorporating this counter into a game, such as a quiz show, can help keep track of scores or time remaining. Players can press button to indicate correct answers or respond to questions.

7. Digital Clock: By expanding the code and adding a real-time clock module, we can convert this project into a simple digital clock that displays the time in hours and minutes.

8. Event Timer: Use the counter to time events or activities in a sports competition, such as counting the number of laps run by athletes or measuring the time elapsed in a race.

9. Traffic Light Simulator: With some modifications, the two-digit counter can simulate traffic light sequences. For example, it can count down the time remaining for a green light before it turns yellow and then red.

10. Attendance System: It can serve as an attendance system for schools or workplaces. Each person can press a button to register their presence, and the system keeps track of the total number of attendees.

11. Temperature and Humidity Monitor: By adding environmental sensors and modifying the code, we can turn this project into a temperature and humidity monitor that displays the values on the 7-segment displays.

12. Fitness Tracker: The project can be used to keep track of fitness-related data, such as the number of steps taken or calories burned in a workout session.

These applications demonstrate the versatility of a two-digit counter using push button and 7-segment displays with Arduino. By adapting the code and components, it can be customized to serve a wide range of purposes, from simple educational tools to practical . solutions for counting and monitoring various events and data.

solutions for counting and monitoring various events and data.

**ALGORITHM:-**

To create a two-digit counter using push button and 7-segment displays with Arduino, we'll need to implement an algorithm in wer Arduino code. This algorithm will handle input from the push button and update the 7-segment displays to show the current count. Here's a step-by-step algorithm for this project:

Initialization:

1. Define the pin connections for the two 7-segment displays and the push button.

2. Set up the pins as inputs for the push button and outputs for the 7-segment displays.

3. Initialize variables to hold the count, such as `count = 0`, and variables to track button states.

Main Loop:

4. Read the state of each push button and store the results in variables (e.g., `buttonIncrementState`, `buttonDecrementState`, and `buttonResetState`).

5. Implement debouncing logic to ensure reliable button input. We can use a software debounce function or use a hardware solution.

6. Check if the increment button is pressed (e.g., `if (buttonIncrementState == HIGH)`). If it's pressed, increment the count by 1.

7. Check if the decrement button is pressed (e.g., `if (buttonDecrementState == HIGH)`). If it's pressed, decrement the count by 1. Ensure the count never goes below 0.

8. Check if the reset button is pressed (e.g., `if (buttonResetState == HIGH)`). If it's pressed, set the count to 0.

9. Update the 7-segment displays to show the current count. This may involve dividing the count into tens and ones digits and using a function to display these digits on the displays.

10. If the count goes beyond 99, consider implementing a rollover logic to reset the count back to 0.

11. Implement multiplexing to display the digits on both 7-segment displays if required.

**Display Logic:**

12. Create a function that maps digits (0-9) to the appropriate segments to display on the 7-segment displays.

13. Implement a function that takes the current count and separates it into tens and ones digits. Then, call the digit mapping function for each digit to display the numbers on the displays.

Debouncing Logic:

14. Implement debouncing logic for the push button to prevent false button presses or counts.

15. A simple software debouncing technique involves checking the state of the button over multiple loops and only considering it pressed when it maintains the same state for a defined period.

Loop Continuation:

16. Continue looping through the main loop, constantly reading the button states, updating the count, and refreshing the displays.

By following this algorithm, we can create a functional two-digit counter using push button and 7-segment displays with Arduino. Make sure to test and calibrate wer project to ensure that it accurately counts and displays numbers as expected.

**FLOWCHART**

This flowchart outlines the main elements of your program:

Initialization: Set up the Arduino and pins, initialize count and button states.

Main Loop: Continuously check the button states, debounce them, and react accordingly.

Debounce Button: Ensure stable button states are detected before processing.

Button Presses: Check if any of the button (increment, decrement, reset) are pressed.

Count Manipulation: If a button is pressed, increment, decrement, or reset the count as required.

Display: Separate the count into tens and ones digits, and update the 7-segment display.

Count Rollover: Check if the count exceeds 99, and if so, consider resetting it.

Continue Loop: Return to the main loop and keep checking for button presses.

This flowchart provides a high-level view of how your program should function. When coding, you'll need to translate each step into the appropriate Arduino code, including the button debouncing process and the display updating logic.

**PROGRAM**

int count = 0; // Declare and initialize a variable 'count' to 0.

void setup() {

Serial.begin(9600); // Initialize serial communication at a baud rate of 9600.

pinMode(A0, INPUT); // Set pin A0 as an input.

for (int i = 0; i <= 13; i++) {

pinMode(i, OUTPUT); // Set pins 0 to 13 as outputs.

}

}

const int number[11] = { // Define an array 'number' with 11 elements.

0b0111111, // Binary representation of digit 0

0b0000110, // Binary representation of digit 1

0b1011011, // Binary representation of digit 2

0b1001111, // Binary representation of digit 3

0b1100110, // Binary representation of digit 4

0b1101101, // Binary representation of digit 5

0b1111101, // Binary representation of digit 6

0b0000111, // Binary representation of digit 7

0b1111111, // Binary representation of digit 8

0b1101111 , // Binary representation of digit 9

};

void loop() {

int buttonState = digitalRead(A0); // Read the state of the button connected to A0.

if (count > 99) {

count = 0; // If count exceeds 99, reset it to 0.

}

if (buttonState == HIGH) {

Serial.println("HIGH"); // Print "HIGH" to the serial monitor.

// Split the count into tens and ones digits.

int tens = count / 10;

int ones = count % 10;

// Special case when count is 0 to avoid leading zero.

if (count == 0) {

tens = 0;

}

// Call functions to display the tens and ones digits on the 7-segment display.

display\_tens(tens);

display\_ones(ones);

count++; // Increment the count.

delay(300); // Delay for 300 milliseconds.

} else {

Serial.println("LOW"); // Print "LOW" to the serial monitor.

}

}

void display\_tens(const int tens) {

int pin1, a;

for (pin1 = 0, a = 0; pin1 < 7; pin1++, a++) {

// Set the digital output of pins 0 to 6 based on the bit pattern for the tens digit.

digitalWrite(pin1, bitRead(number[tens], a));

}

}

void display\_ones(const int x) {

int pin2, b;

for (pin2 = 7, b = 0; pin2 <= 13; pin2++, b++) {

// Set the digital output of pins 7 to 13 based on the bit pattern for the ones digit.

digitalWrite(pin2, bitRead(number[x], b));

}

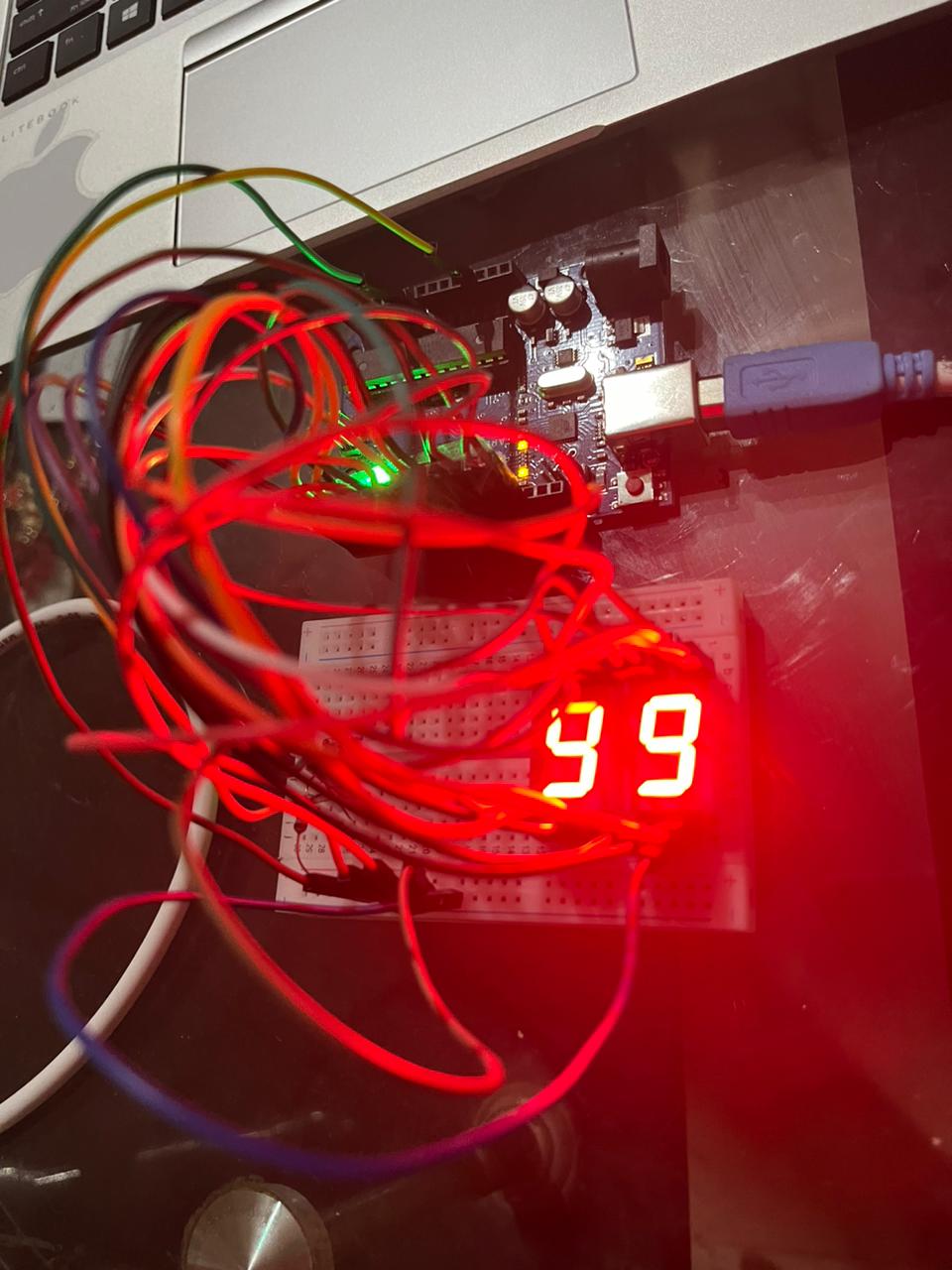
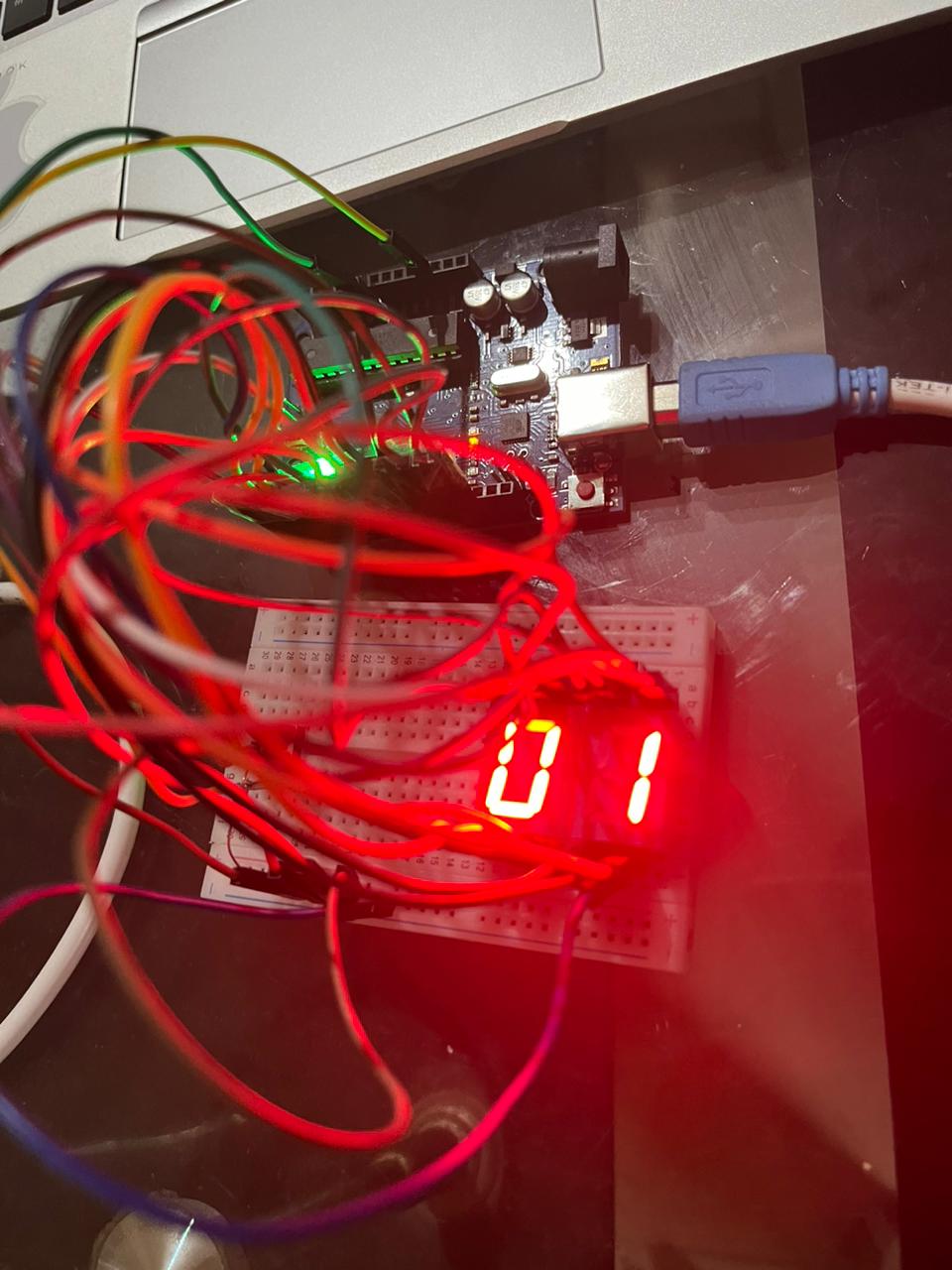
}

**Circuit Diagram**

**A circuit board with wires

Description automatically generated**

**OUTPUT**

****

# CONCLUSION

The "Two Digit Counter Using Arduino & 7 Segment Display & Push Button" project is a great way to learn about microcontroller programming, interfacing digital displays, and using push buttons for user input. As we conclude this project, let's recap some key takeaways and potential future improvements:

**Learning Experience**: This project provides an excellent opportunity to understand the basics of Arduino programming. We learned how to control a 7-segment display and handle user input with push buttons. This foundational knowledge can be expanded for more complex projects in the future.

**Digital Display**: We successfully interfaced a common anode 7-segment display with the Arduino. This skill can be extended to create more advanced displays for different applications, such as timers, temperature readouts, or scoreboards.

**User Interaction**: The push buttons allowed for user interaction, which is essential in various projects. Understanding how to debounce buttons and handle multiple inputs can be crucial for other applications like menu systems or game controllers.

**Counting and Logic**: This project involved counting and simple logic for displaying two-digit numbers. These concepts can be be applied to a wide range of applications, such as digital counters, timers, and digital scoreboards.

**Integration**: Think about how this project can be integrated into other systems. For instance, we can use it as a part of a larger automation system or as a component of a game or puzzle.

# COURSE OUTCOME

The successful completion of this project has significantly contributed to the fulfillment of the following course outcomes.

* CO-III : Select & use the appropriate ESP module / Arduino for real world application

We have learned how to implement and design a Two Digit Counter Using Arduino & 7 Segment Display & Push Button



**References:-**

1**.**<https://www.instructables.com/2-Digit-Seven-Segment-Display/>

2.<https://microdigisoft.com/2-digit-7-segment-counter00-99with-arduino/>

3.<https://forum.arduino.cc/t/2-digits-7-segment-display/863069>

4.<https://www.electronics-tutorials.ws/blog/7-segment-display-tutorial.html>



fhhhtg