**7-segment display circuit project**.

Code

// Define the segment pins for the 7-segment display

int a = 2; // Segment a

int b = 3; // Segment b

int c = 4; // Segment c

int d = 5; // Segment d

int e = 6; // Segment e

int f = 7; // Segment f

int g = 8; // Segment g

// Function to display a number (0-9) on a 7-segment display

void displayDigit(int num)

{

switch (num)

{

case 0: // Display 0

digitalWrite(a, HIGH);

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(d, HIGH);

digitalWrite(e, HIGH);

digitalWrite(f, HIGH);

digitalWrite(g, LOW);

break;

case 1: // Display 1

digitalWrite(a, LOW);

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(d, LOW);

digitalWrite(e, LOW);

digitalWrite(f, LOW);

digitalWrite(g, LOW);

break;

case 2: // Display 2

digitalWrite(a, HIGH);

digitalWrite(b, HIGH);

digitalWrite(c, LOW);

digitalWrite(d, HIGH);

digitalWrite(e, HIGH);

digitalWrite(f, LOW);

digitalWrite(g, HIGH);

break;

case 3: // Display 3

digitalWrite(a, HIGH);

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(d, HIGH);

digitalWrite(e, LOW);

digitalWrite(f, LOW);

digitalWrite(g, HIGH);

break;

case 4: // Display 4

digitalWrite(a, LOW);

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(d, LOW);

digitalWrite(e, LOW);

digitalWrite(f, HIGH);

digitalWrite(g, HIGH);

break;

case 5: // Display 5

digitalWrite(a, HIGH);

digitalWrite(b, LOW);

digitalWrite(c, HIGH);

digitalWrite(d, HIGH);

digitalWrite(e, LOW);

digitalWrite(f, HIGH);

digitalWrite(g, HIGH);

break;

case 6: // Display 6

digitalWrite(a, HIGH);

digitalWrite(b, LOW);

digitalWrite(c, HIGH);

digitalWrite(d, HIGH);

digitalWrite(e, HIGH);

digitalWrite(f, HIGH);

digitalWrite(g, HIGH);

break;

case 7: // Display 7

digitalWrite(a, HIGH);

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(d, LOW);

digitalWrite(e, LOW);

digitalWrite(f, LOW);

digitalWrite(g, LOW);

break;

case 8: // Display 8

digitalWrite(a, HIGH);

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(d, HIGH);

digitalWrite(e, HIGH);

digitalWrite(f, HIGH);

digitalWrite(g, HIGH);

break;

case 9: // Display 9

digitalWrite(a, HIGH);

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(d, HIGH);

digitalWrite(e, LOW);

digitalWrite(f, HIGH);

digitalWrite(g, HIGH);

break;

}

}

void setup()

{

// Set the segment pins as output

pinMode(a, OUTPUT);

pinMode(b, OUTPUT);

pinMode(c, OUTPUT);

pinMode(d, OUTPUT);

pinMode(e, OUTPUT);

pinMode(f, OUTPUT);

pinMode(g, OUTPUT);

}

void loop() {

// Loop through digits 0 to 9 and display each for 1 second

for (int i = 0; i <= 9; i++)

{

displayDigit(i);

delay(1000); // Wait 1 second before displaying the next number

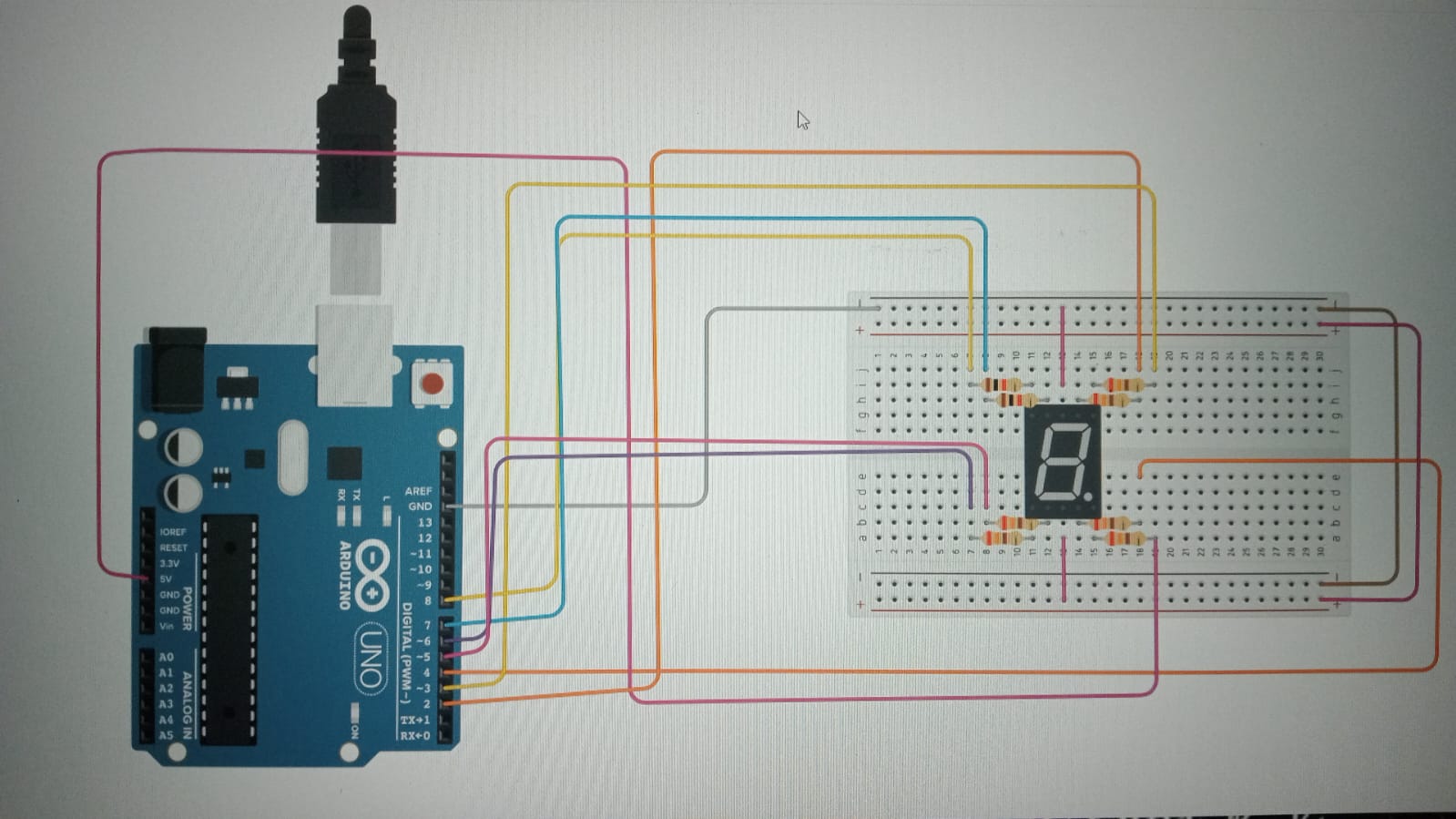
}

}

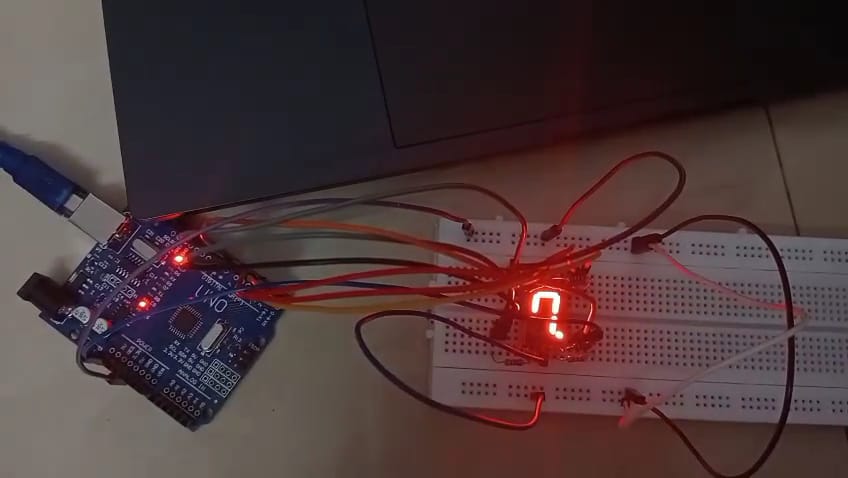
Helper function

Helper function here is c programming .

Diagrams/



Images/ project-photo 1



Docs/

Components of an7 Segment Display (0-9) numbers

1. Arduino board 2. 7-segment display (common cathode) 3. 7 current-limiting resistors (220Ω) 4. Connecting wires 5. Breadboard

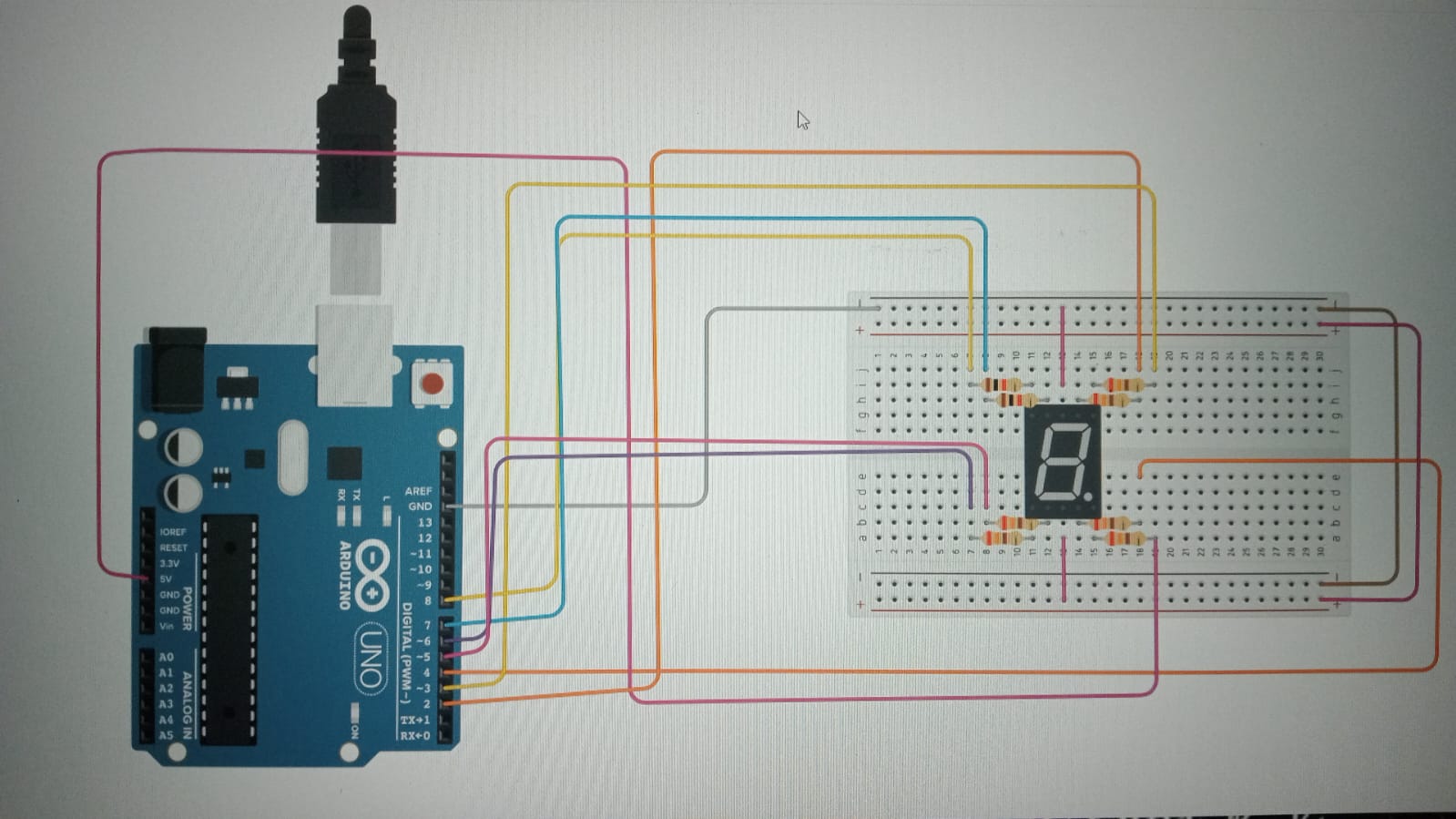
**1. Project Overview**

* **Introduction:**
  + This project demonstrates how to control a 7-segment display using an Arduino to display numbers 0-9 in a loop.
* **Goal:**
  + The goal is to create a simple digital counter using a 7-segment display controlled by an Arduino.

**2. Components List**

* **List all components required for the project:**
  + **7-Segment Display** (1 or more depending on the design)
  + **Arduino** (e.g., Arduino Uno)
  + **Resistors** (e.g., 220Ω or 330Ω, one per segment)
  + **Jumper wires**
  + **Breadboard**
  + **Power source** (e.g., USB cable or external power supply)
  + **Other components** (optional, e.g., push buttons for input, etc.)

1. **Circuit Diagram**



* **Show the schematic:**
* **Connection**

This image shows a circuit diagram involving an Arduino Uno, a 7-segment display, and a breadboard. Below is the detailed explanation of how this setup works and how the connections are made.

---

Connections

1. 7-Segment Display Pins

A 7-segment display has 8 LED segments: a, b, c, d, e, f, g, and the decimal point (dp).

Each segment lights up based on the voltage applied to its pin.

For a common cathode 7-segment display:

The common cathode pin is connected to GND (Ground).

The segments (a, b, c, d, e, f, g) are connected to the respective Arduino pins.

2. Arduino to 7-Segment Display Wiring Assuming the 7-segment display is common cathode:

Connect the GND pin on the 7-segment display to the GND pin of the Arduino.

Connect the a pin to a digital pin (e.g., Pin 2 on Arduino Uno).

Connect the b pin to another digital pin (e.g., Pin 3).

Continue this for all segments (c, d, e, f, g) to Arduino pins.

If the decimal point (dp) is used, connect it to another pin (optional).

3. Current Limiting Resistors

Place a 220-ohm resistor between each segment pin and the Arduino digital pins to protect the LED segments from high current.

4. Power Supply

Ensure the Arduino Uno is powered via USB or an external power source.

Code Explanation

**Pin Definition**

cpp

Copy code

int a = 2; // Segment a

int b = 3; // Segment b

int c = 4; // Segment c

int d = 5; // Segment d

int e = 6; // Segment e

int f = 7; // Segment f

int g = 8; // Segment g

* The **7-segment display** has seven segments (labeled a through g), and each segment is controlled by a digital pin on the Arduino.
* The variables a, b, c, d, e, f, and g are assigned to Arduino pins 2 to 8, respectively. These will be used to control each segment of the display.

**2. Function to Display a Digit**

cpp

Copy code

void displayDigit(int num)

{

switch (num)

{

case 0: // Display 0

digitalWrite(a, HIGH);

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(d, HIGH);

digitalWrite(e, HIGH);

digitalWrite(f, HIGH);

digitalWrite(g, LOW);

break;

case 1: // Display 1

digitalWrite(a, LOW);

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(d, LOW);

digitalWrite(e, LOW);

digitalWrite(f, LOW);

digitalWrite(g, LOW);

break;

case 2: // Display 2

digitalWrite(a, HIGH);

digitalWrite(b, HIGH);

digitalWrite(c, LOW);

digitalWrite(d, HIGH);

digitalWrite(e, HIGH);

digitalWrite(f, LOW);

digitalWrite(g, HIGH);

break;

// Add similar cases for 3 through 9

}

}

* **Purpose:** This function (displayDigit) takes an integer (num) as input (ranging from 0 to 9) and turns on/off specific segments of the 7-segment display to display the corresponding digit.
* **How it works:**
  + A **switch statement** is used to check the value of num and execute the appropriate case for each digit (0-9).
  + Each case corresponds to a number (0-9), and inside each case, the **digitalWrite()** function is used to turn on (HIGH) or off (LOW) the individual segments of the 7-segment display.
  + For example:
    - **Case 0**: For displaying the number 0, all segments except g are lit up (set to HIGH).
    - **Case 1**: For displaying the number 1, only segments b and c are lit up.
  + **Key Concept:** The 7-segment display is **a shared common cathode or anode** display, meaning turning segments on or off will depend on whether the segment is active high or low. The pattern of HIGH or LOW in each case matches the typical wiring and logic for a common cathode 7-segment display (where LOW turns the segment on, and HIGH turns it off).

**3. Setup Function**

cpp

Copy code

void setup()

{

// Set the segment pins as output

pinMode(a, OUTPUT);

pinMode(b, OUTPUT);

pinMode(c, OUTPUT);

pinMode(d, OUTPUT);

pinMode(e, OUTPUT);

pinMode(f, OUTPUT);

pinMode(g, OUTPUT);

}

* **Purpose:** The setup() function is called once when the Arduino program starts.
* **Pin Configuration:**
  + This function sets all the segment pins (a to g) as **output** using the **pinMode()** function, which ensures that the pins can send signals (HIGH or LOW) to control the display.
  + Without this, the Arduino wouldn't know that these pins need to output signals to the segments of the display.

**4. Loop Function**

cpp

Copy code

void loop() {

// Loop through digits 0 to 9 and display each for 1 second

for (int i = 0; i <= 9; i++)

{

displayDigit(i); // Display the digit

delay(1000); // Wait 1 second before displaying the next number

}

}

* **Purpose:** The loop() function runs continuously after the setup() function is executed. This is where the main logic happens.
* **How it works:**
  + The for loop runs from i = 0 to i = 9, which means it will display numbers 0 through 9.
  + Inside the loop, the **displayDigit(i)** function is called, which shows the current value of i on the 7-segment display.
  + After displaying each digit, the **delay(1000)** function pauses the program for **1 second** (1000 milliseconds), so the displayed number stays on the screen for 1 second before moving to the next number.

**Code Summary:**

* **Pin Setup:** Arduino pins 2-8 are connected to the segments a to g of the 7-segment display.
* **displayDigit() Function:** A switch statement turns on/off the appropriate segments to display the numbers 0-9.
* **setup() Function:** Initializes all pins controlling the segments as outputs.
* **loop() Function:** Loops through digits 0 to 9, displaying each one for 1 second with a delay between them.

Summary

This circuit allows the Arduino to control which segments light up on the 7-segment display, displaying numbers or symbols. Ensure proper connections, and verify whether the 7-segment display is common cathode or common anode to adjust your connections accordingly.

**Total Cost Estimate :**

* **7-Segment Display: ₹40 (for one display)**
* **Arduino Board: ₹350 (for Arduino Uno clone)**
* **Resistors: ₹10 (for a pack)**
* **Jumper Wires: ₹150 (for a pack)**
* **Breadboard : ₹250**

**Project Overview:**

**This project is designed to display numbers (0 to 9) on a 7-segment display using an Arduino board. The Arduino will control the individual segments of the display, turning them on and off to form the numbers. The display will cycle through the digits 0 to 9, displaying each digit for 1 second. This project is ideal for beginners who want to learn how to control a 7-segment display using an Arduino.**

**Working Principle:**

* **The 7-segment display consists of seven segments labeled a, b, c, d, e, f, g. By turning certain segments on or off, we can display digits from 0 to 9. Each digit corresponds to a specific combination of segments being lit.**
* **The Arduino Uno controls these segments by sending HIGH (5V) or LOW (0V) signals to each pin connected to the segments. The segments are controlled using digital pins of the Arduino.**
* **Resistors are used to limit the current passing through the segments to prevent damage. Each segment of the display will be connected to a digital output pin of the Arduino, and appropriate resistors are placed in series with each segment to avoid excessive current flow.**
* **The loop in the Arduino code continuously cycles through the digits 0 to 9. Each number is displayed for 1 second, and then the next number appears. The Arduino uses the digitalWrite() function to control which segments are lit.**

**Applications:**

* **Digital Clocks: You can extend this project to make a digital clock by adding timekeeping features.**
* **Counters: Useful in creating counters for various applications like traffic counters, device counters, etc.**
* **Learning Platform: This project helps in learning the basics of Arduino and digital electronics, especially controlling displays and using basic programming.**

**Conclusion:**

**This 7-segment display counter is a great introductory project for anyone getting started with Arduino. It teaches how to control a 7-segment display, use a switch-case structure in programming, and understand how to wire components in an Arduino circuit.**