**Step 1: Create a Tinkercad Account**

- Go to Tinkercad and create an account or log in.

**Step 2: Create a New Circuit**

1. Click on **"Circuits"** from the left menu.
2. Click on **"Create new Circuit."**

**Step 3: Add Components**

1. **Arduino Uno**: Drag and drop the Arduino Uno onto the workspace.
2. **7-Segment Display**: Search for "7-segment" in the components panel and drag it onto the workspace.
3. **Resistors**: Add 7 resistors (220 ohms is typical) to limit current to the segments.
4. **Breadboard**: Optional, but can help with organizing connections.

**Step 4: Connect the 7-Segment Display**

Here's how to connect the components:

**Pinout (Common Cathode Display)**

Assuming you have a common cathode display, the typical pin configuration is:

- **A**: Pin 1
- **B**: Pin 2
- **C**: Pin 3
- **D**: Pin 4
- **E**: Pin 5
- **F**: Pin 6
- **G**: Pin 7
- **Common Cathode**: Pin 8

**Connections**

1. Connect each segment (A to G) to a digital pin on the Arduino through a resistor:
    - A (Pin 1) to **Pin 2** on Arduino
    - B (Pin 2) to **Pin 3** on Arduino
    - C (Pin 3) to **Pin 4** on Arduino
    - D (Pin 4) to **Pin 5** on Arduino
    - E (Pin 5) to **Pin 6** on Arduino
    - F (Pin 6) to **Pin 7** on Arduino
    - G (Pin 7) to **Pin 8** on Arduino
2. Connect the common cathode (Pin 8) to **GND** on the Arduino.

**Step 5: Write the Code**

1. Click on **"Code"** in the top right corner of the workspace.
2. Change the code editor to **"Blocks"** or **"Text"** as per your preference.
3. If using text, paste the following code:

cpp
Copy code

```
// Pin assignments
const int A = 2;
const int B = 3;
const int C = 4;
const int D = 5;
const int E = 6;
const int F = 7;
const int G = 8;

// Array to represent numbers 0-9
const int digits[10][7] = {
  {1, 1, 1, 1, 1, 1, 0}, // 0
  {0, 1, 1, 0, 0, 0, 0}, // 1
  {1, 1, 0, 1, 1, 0, 1}, // 2
```

```
  {1, 1, 1, 1, 0, 0, 1}, // 3
  {0, 1, 1, 0, 0, 1, 1}, // 4
  {1, 0, 1, 1, 0, 1, 1}, // 5
  {1, 0, 1, 1, 1, 1, 1}, // 6
  {1, 1, 1, 0, 0, 0, 0}, // 7
  {1, 1, 1, 1, 1, 1, 1}, // 8
  {1, 1, 1, 1, 0, 1, 1}  // 9
};

void setup() {
  // Set 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() {
  for (int num = 0; num < 10; num++) {
    displayNumber(num);
    delay(1000); // Delay to see each number
  }
}

void displayNumber(int num) {
  digitalWrite(A, digits[num][0]);
  digitalWrite(B, digits[num][1]);
  digitalWrite(C, digits[num][2]);
```

```
  digitalWrite(D, digits[num][3]);
  digitalWrite(E, digits[num][4]);
  digitalWrite(F, digits[num][5]);
  digitalWrite(G, digits[num][6]);
}
```

**Step 6: Simulate**

1. Click on **"Start Simulation"** in the top right corner to run the code.
2. You should see the numbers 0 to 9 displayed sequentially on the 7-segment display.

**Tips**

- If the display doesn't light up, check your connections and make sure all pins are connected correctly.
- Adjust the delay in the loop to change how fast the numbers appear.

Now you have a working 7-segment display connected to an Arduino in Tinke

Code

```
// Pin assignments

const int A = 2;

const int B = 3;

const int C = 4;

const int D = 5;

const int E = 6;

const int F = 7;

const int G = 8;
```

```
// Array to represent numbers 0-9
const int digits[10][7] = {
  {1, 1, 1, 1, 1, 1, 0}, // 0
  {0, 1, 1, 0, 0, 0, 0}, // 1
  {1, 1, 0, 1, 1, 0, 1}, // 2
  {1, 1, 1, 1, 0, 0, 1}, // 3
  {0, 1, 1, 0, 0, 1, 1}, // 4
  {1, 0, 1, 1, 0, 1, 1}, // 5
  {1, 0, 1, 1, 1, 1, 1}, // 6
  {1, 1, 1, 0, 0, 0, 0}, // 7
  {1, 1, 1, 1, 1, 1, 1}, // 8
  {1, 1, 1, 1, 0, 1, 1}  // 9
};

void setup() {
  // Set 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() {
  for (int num = 0; num < 10; num++) {
    displayNumber(num);
    delay(1000); // Delay to see each number
  }
}

void displayNumber(int num) {
  digitalWrite(A, digits[num][0]);
  digitalWrite(B, digits[num][1]);
  digitalWrite(C, digits[num][2]);
  digitalWrite(D, digits[num][3]);
  digitalWrite(E, digits[num][4]);
  digitalWrite(F, digits[num][5]);
  digitalWrite(G, digits[num][6]);
}
```

**Code with comments**

```
// Pin assignments for the segments of the 7-segment display

const int A = 2; // Pin connected to segment A

const int B = 3; // Pin connected to segment B

const int C = 4; // Pin connected to segment C

const int D = 5; // Pin connected to segment D

const int E = 6; // Pin connected to segment E

const int F = 7; // Pin connected to segment F

const int G = 8; // Pin connected to segment G


// Array to represent numbers 0-9 in 7-segment display format

const int digits[10][7] = {

  {1, 1, 1, 1, 1, 1, 0}, // 0: All segments on except G

  {0, 1, 1, 0, 0, 0, 0}, // 1: Only segments B and C on

  {1, 1, 0, 1, 1, 0, 1}, // 2: Segments A, B, D, E, and G on

  {1, 1, 1, 1, 0, 0, 1}, // 3: Segments A, B, C, D, and G on

  {0, 1, 1, 0, 0, 1, 1}, // 4: Segments B, C, F, and G on

  {1, 0, 1, 1, 0, 1, 1}, // 5: Segments A, C, D, F, and G on

  {1, 0, 1, 1, 1, 1, 1}, // 6: All segments on except B
```

```
  {1, 1, 1, 0, 0, 0, 0}, // 7: Segments A, B, and C on
  {1, 1, 1, 1, 1, 1, 1}, // 8: All segments on
  {1, 1, 1, 1, 0, 1, 1}  // 9: Segments A, B, C, D, F, and G on
};

void setup() {
  // Initialize each segment pin as OUTPUT
  pinMode(A, OUTPUT); // Set pin A as output
  pinMode(B, OUTPUT); // Set pin B as output
  pinMode(C, OUTPUT); // Set pin C as output
  pinMode(D, OUTPUT); // Set pin D as output
  pinMode(E, OUTPUT); // Set pin E as output
  pinMode(F, OUTPUT); // Set pin F as output
  pinMode(G, OUTPUT); // Set pin G as output
}

void loop() {
  // Loop through numbers 0 to 9
  for (int num = 0; num < 10; num++) {
    displayNumber(num); // Call function to display the number
    delay(1000); // Wait for 1 second to see each number
```

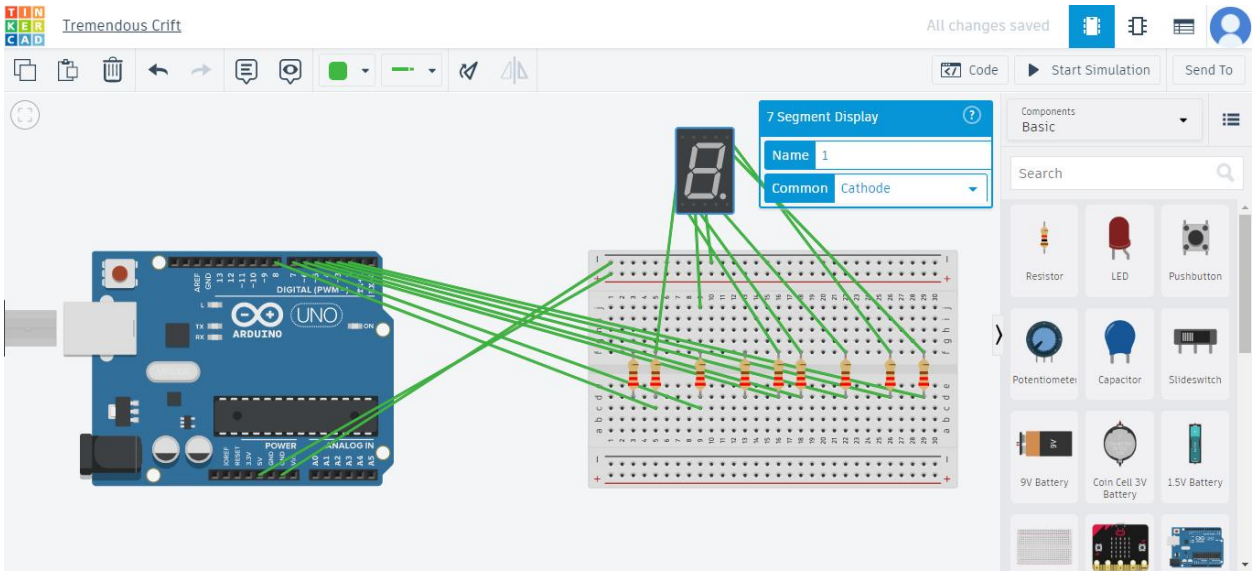
```
  }
}

// Function to display a number on the 7-segment display
void displayNumber(int num) {
  // Set each segment based on the number's representation in the array
  digitalWrite(A, digits[num][0]); // Control segment A
  digitalWrite(B, digits[num][1]); // Control segment B
  digitalWrite(C, digits[num][2]); // Control segment C
  digitalWrite(D, digits[num][3]); // Control segment D
  digitalWrite(E, digits[num][4]); // Control segment E
  digitalWrite(F, digits[num][5]); // Control segment F
  digitalWrite(G, digits[num][6]); // Control segment G
}
```

**Explanation**

- Each segment of the 7-segment display is controlled by a digital pin on the Arduino.
- The digits array defines which segments should be lit for each digit from 0 to 9.
- In the setup() function, we configure each pin as an output.
- The loop() function displays each number in sequence, pausing for one second between numbers.
- The displayNumber() function turns on or off the appropriate segments for the number being displayed.

Common cathode




Raspberry pi code

**Wiring the 7-Segment Display**

First, let's assign the GPIO pins on the Raspberry Pi. Here's a sample pin diagram that corresponds to the Arduino pins you provided:

- **A**: GPIO 17
- **B**: GPIO 18
- **C**: GPIO 27
- **D**: GPIO 22
- **E**: GPIO 23
- **F**: GPIO 24
- **G**: GPIO 25

**Python Code**

You can install the RPi.GPIO library if you haven't already:

pip install RPi.GPIO

**Here's the Python code:**

```
import RPi.GPIO as GPIO
import time

# Pin assignments for the segments of the 7-segment display
A = 17  # Pin connected to segment A
B = 18  # Pin connected to segment B
C = 27  # Pin connected to segment C
D = 22  # Pin connected to segment D
E = 23  # Pin connected to segment E
F = 24  # Pin connected to segment F
G = 25  # Pin connected to segment G

# Array to represent numbers 0-9 in 7-segment display format
digits = [
    [1, 1, 1, 1, 1, 1, 0],  # 0
    [0, 1, 1, 0, 0, 0, 0],  # 1
    [1, 1, 0, 1, 1, 0, 1],  # 2
    [1, 1, 1, 1, 0, 0, 1],  # 3
    [0, 1, 1, 0, 0, 1, 1],  # 4
    [1, 0, 1, 1, 0, 1, 1],  # 5
    [1, 0, 1, 1, 1, 1, 1],  # 6
    [1, 1, 1, 0, 0, 0, 0],  # 7
    [1, 1, 1, 1, 1, 1, 1],  # 8
    [1, 1, 1, 1, 0, 1, 1],  # 9
]
```

```
# Setup GPIO
GPIO.setmode(GPIO.BCM)
segment_pins = [A, B, C, D, E, F, G]
for pin in segment_pins:
    GPIO.setup(pin, GPIO.OUT)

def display_number(num):
    # Set each segment based on the number's representation in the array
    for i in range(7):
        GPIO.output(segment_pins[i], digits[num][i])

try:
    # Loop through numbers 0 to 9
    while True:
        for num in range(10):
            display_number(num)  # Display the number
            time.sleep(1)        # Wait for 1 second

except KeyboardInterrupt:
    pass
finally:
    GPIO.cleanup()  # Cleanup GPIO settings
```

**Explanation**

- **GPIO Setup**: The GPIO pins are set up to use the BCM numbering scheme.
- **Display Function**: The display_number function controls the segments based on the number you want to display.
- **Main Loop**: The main loop continuously displays numbers from 0 to 9 with a 1-second delay between each.

**Running the Code**

1. Save the code to a file named seven_segment_display.py.
2. Run the script using:

   python3 seven_segment_display.py

3. To stop the program, use Ctrl + C. This will trigger the cleanup to reset the GPIO pins.

The code you provided is designed for a **common cathode** 7-segment display. In a common cathode configuration, the cathodes of the segments are connected to ground, and you apply a HIGH signal to the segment pins to turn them on.

**How to Modify for Common Anode**

If you were to use a **common anode** display instead, you would need to invert the logic in the display_digit function. You can do this by changing:

**GPIO.output(segments[i], digits[num][i])**

to:

**GPIO.output(segments[i], not digits[num][i])**

**Summary**

- **Common Cathode**: Code as-is (HIGH turns segments on).
- **Common Anode**: Invert the logic as shown above (LOW turns segments on).