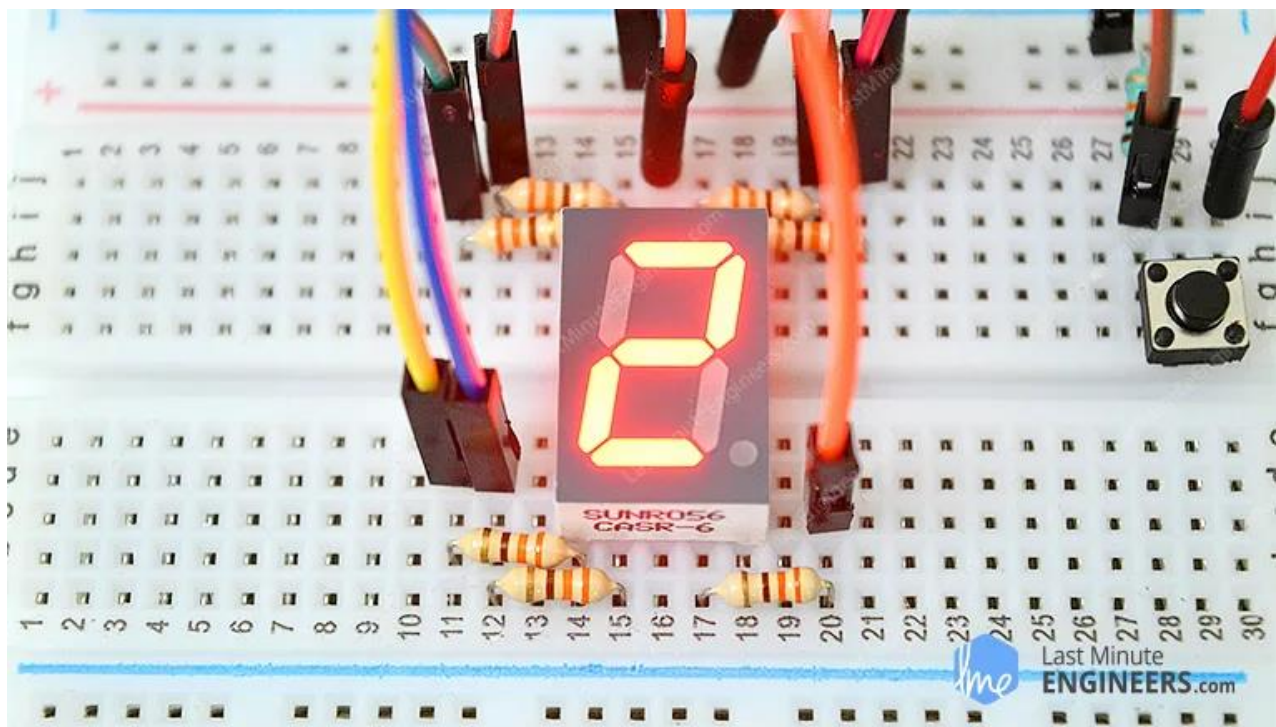


# SEVEN SEGMENT

Have you ever noticed those tense scenes in classic movies where the hero is desperately trying to defuse a bomb before it destroys the entire city? As the hero anxiously watches the timer counting down, each passing second becomes more precious than the last. If you pay close attention to these scenes, you'll realize that almost all of these movies use seven-segment displays for their countdown timers. It makes perfect sense – how else would our heroes know exactly how much time they have left to save the day?

The same technology that helped our heroes also appears in everyday devices all around us—from microwave timers and alarm clocks to car dashboards and elevator floor indicators. And why wouldn't it be: they're easy to use, affordable, and simple to read in both bright and dim lighting.

In this tutorial, we'll learn how to use a 7-segment display with an Arduino. So grab your Arduino and 7-segment display, and let's get started!

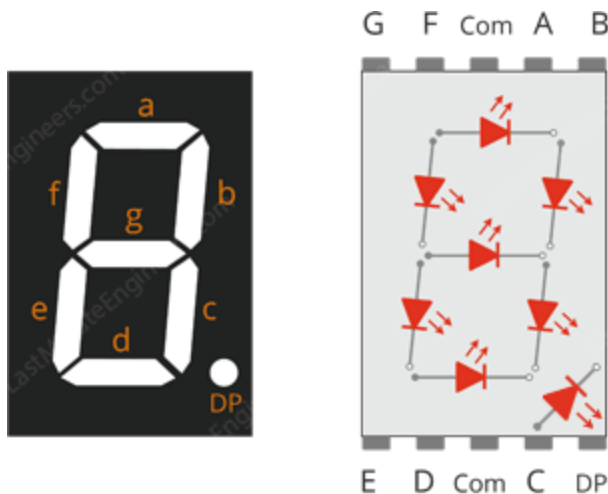


## THE 7-SEGMENT DISPLAY

While a 7-segment display looks like one complete unit, it's actually made up of seven separate LEDs (Light Emitting Diodes) arranged in the shape of the number "8". Each of these LEDs is called a segment. There's often an eighth segment too – the decimal point (DP) – which lets you display decimal numbers.

These LEDs work together to create numbers and some letters, but their internal connections are important to understand.

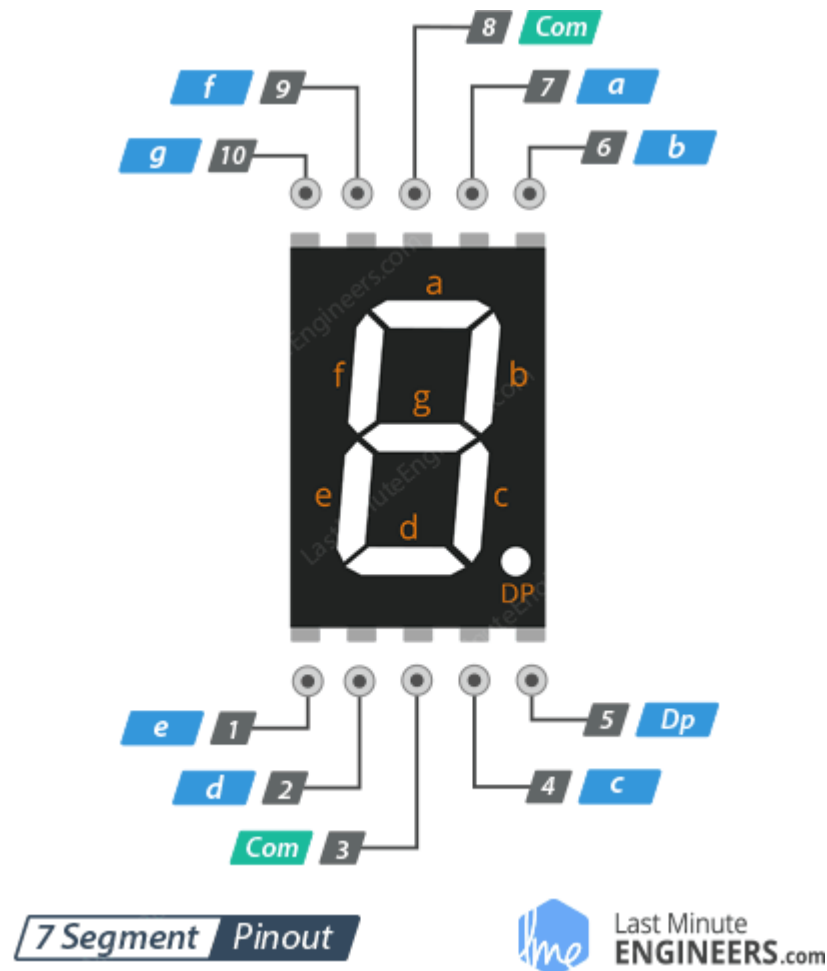
Each segment has two electrical connection points, just like any standard LED. However, only one of these connection points extends outside the plastic casing as an individual pin that you can connect to. These pins are labeled with letters from 'a' through 'g'. The other connection points from all seven LEDs don't get their own individual pins. Instead, these connection points are all wired together inside the plastic case to form a single shared connection called the common pin (COM).



You can control each segment individually by turning its pin on (HIGH) or off (LOW), just like you would with any standard LED. By lighting up different combinations of segments, you can display all the numbers from 0 through 9. With a bit of creativity, you can even show some letters of the alphabet!

## 7-SEGMENT DISPLAY PINOUT

Let's look at how the segments are arranged so you can understand which pin controls which segment:



a, b, c, d, e, f, g, and DP pins each control one segment of the display. By turning on just the right segments, you can create any number you need.

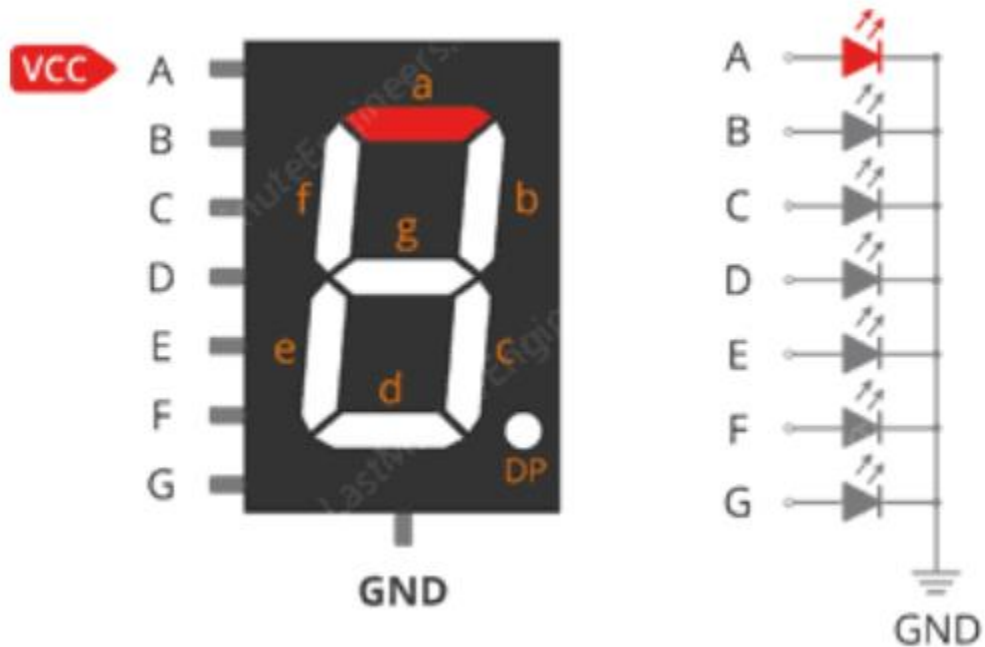
COM pins (usually pins 3 and 8) are connected together inside the display. Depending on what type of display you have, you'll need to connect this pin to either ground (if you have a common cathode display) or 5V power (if you have a common anode display). This common (COM) connection completes the circuit for all the segments, allowing them to light up when activated.

Common Cathode(CC) vs Common Anode(CA)

There are two main types of seven-segment displays: common cathode (CC) and common anode (CA). While they look almost identical from the outside, they work differently on the inside because of how their LEDs are connected.

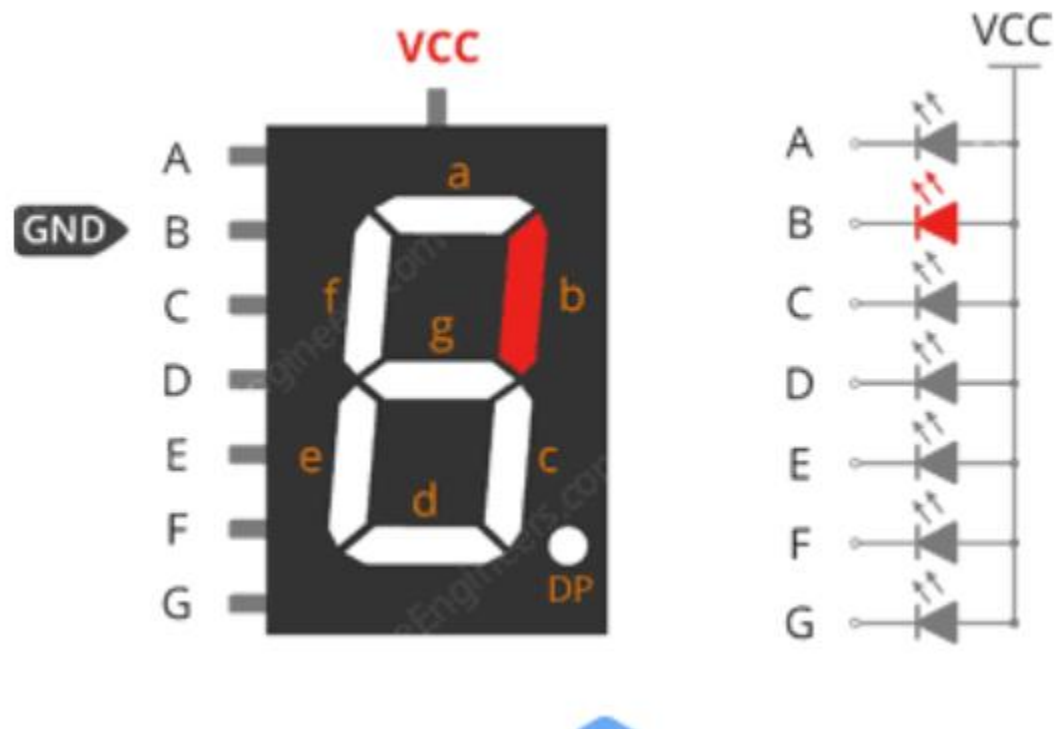
## COMMON CATHODE (CC) DISPLAYS

In a common cathode display, all the negative terminals (cathodes) of the segment LEDs are connected together. To make this type of display work, the common pin (COM) is connected to the ground (GND), and each segment is controlled by applying a positive voltage to its corresponding pin.



## COMMON ANODE (CA) DISPLAYS

In a common anode display, all the positive terminals (anodes) of the segment LEDs are connected together. To make this type of display work, the common pin (COM) is connected to a positive voltage (VCC), and each segment is controlled by applying a ground (GND) or low voltage to its corresponding pin.



## 7-SEGMENT DISPLAY WORKING

A 7-segment display works by lighting up specific combinations of segments to create numbers and some letters. For example:

- To display the number "8," you need to turn on all seven segments (A, B, C, D, E, F, and G)
- To display the number "1," you only need to turn on two segments (B and C)
- To display the letter "A," you would turn on segments A, B, C, E, F, and G

By turning specific segments on or off, you can create any digit from 0 to 9 and even some letters from A to F, as shown in the diagram below.



The truth tables below for 7-segment displays show exactly which segments need to be turned on or off to display each character. By following these patterns in the truth tables, you can program a microcontroller to display any number or basic letter on your 7-segment display.

**Common Cathode Seven Segment Truth-Table**

	a	b	c	d	e	f	g
0	On	On	On	On	On	On	Off
1	Off	On	On	Off	Off	Off	Off
2	On	On	Off	On	On	Off	On
3	On	On	On	On	Off	Off	On
4	Off	On	On	Off	Off	On	On
5	On	Off	On	On	Off	On	On
6	On	Off	On	On	On	On	On
7	On	On	Off	Off	Off	Off	Off
8	On	On	On	On	On	On	On
9	On	On	On	Off	Off	On	On
A	On	On	On	Off	On	On	On
B	Off	Off	On	On	On	On	On
C	On	Off	Off	On	On	On	Off
D	Off	On	On	On	On	Off	On
E	On	Off	Off	On	On	On	On
F	On	Off	Off	Off	On	On	On

**Common Anode Seven Segment Truth-Table**

	a	b	c	d	e	f	g
0	Off	Off	Off	Off	Off	Off	On
1	On	Off	Off	On	On	On	On
2	Off	Off	On	Off	Off	On	Off
3	Off	Off	Off	Off	On	On	Off
4	On	Off	Off	On	On	Off	Off
5	Off	On	Off	Off	On	Off	Off
6	Off	On	Off	Off	Off	Off	Off
7	Off	Off	Off	On	On	On	On
8	Off	Off	Off	Off	Off	Off	Off
9	Off	Off	Off	On	On	Off	Off
A	Off	Off	Off	On	Off	Off	Off
B	On	On	Off	Off	Off	Off	Off
C	Off	On	On	Off	Off	Off	On
D	On	Off	Off	Off	Off	On	Off
E	Off	On	On	Off	Off	Off	Off
F	Off	On	On	On	Off	Off	Off

An important thing to note is that common anode and common cathode displays have opposite electrical connections, so the truth table for a common anode display is the exact opposite of the truth table for a common cathode display.

## SELECTING A CURRENT-LIMITING RESISTOR

As you know, a 7-segment display is actually made up of seven individual LEDs packed together. Since LEDs can only handle a limited amount of electric current, too much current can cause them to burn out quickly. To prevent this, each segment needs its own current-limiting resistor.

For a typical red 7-segment display, each segment works best with about 15 milliamps (mA) of current. To calculate the correct resistor value for a 5-volt circuit, we use this formula:

$$\text{Resistor value} = \frac{\text{Supply voltage} - \text{LED voltage}}{\text{Current}}$$

Given the values:

- Supply voltage = 5 volts
- LED voltage drop = about 2 volts
- Desired current = 15 mA (or 0.015 amps)

Now, plugging these numbers into the formula:

$$\begin{aligned}\text{Resistor value} &= \frac{5V - 2V}{15\text{mA}} \\ &= \frac{3V}{0.015A} \\ &= 200 \Omega\end{aligned}$$

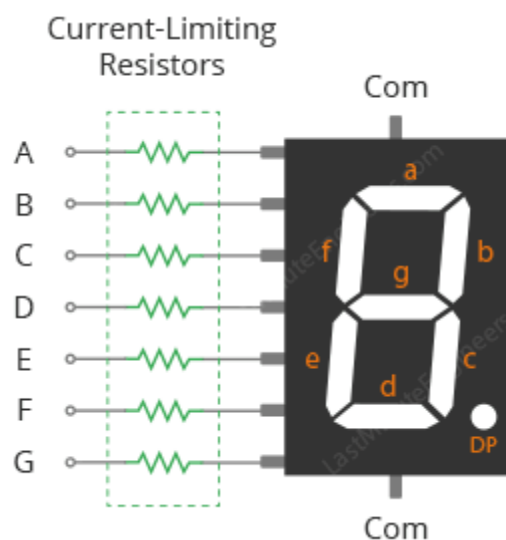
Since resistors come in standard values, we'd round up to 220 ohms.

Using a 220-ohm resistor will make the display slightly dimmer than maximum brightness, which is usually a good thing. Seven-segment displays are typically quite bright, and running them a little dimmer often makes them easier to read and extends their life.

If you need maximum brightness (like for outdoor use), you could use a 150-ohm resistor instead, but this will push the LEDs closer to their limits.

If you're using a different color display and aren't sure about the current requirements, a 330-ohm resistor is a safe starting point. It will limit the current enough to protect the LEDs while still providing reasonable brightness.

The resistors connect between your control circuit (like an Arduino or other microcontroller) and each segment pin of the display, forming a protective barrier that allows the display to work properly without burning out



## WIRING A 7-SEGMENT DISPLAY TO AN ARDUINO

Now that we understand how a 7-segment display works, let's learn how to connect it to an Arduino! This is where we put our knowledge into practice.

Start by placing your 7-segment display on the breadboard so each side of the display sits on opposite sides of the center divider. Position the display with the decimal point facing downward. The pins are numbered in a specific pattern. On the bottom row, you'll find pins 1 through 5 going from left to right. On the top row, you'll find pins 10 through 6, also going from left to right.

Depending on your display type, you'll need to make different connections. For a common anode display, connect one of the COM pins to the Arduino's 5V pin. For a common cathode display, connect the COM pin to the Arduino's GND pin.



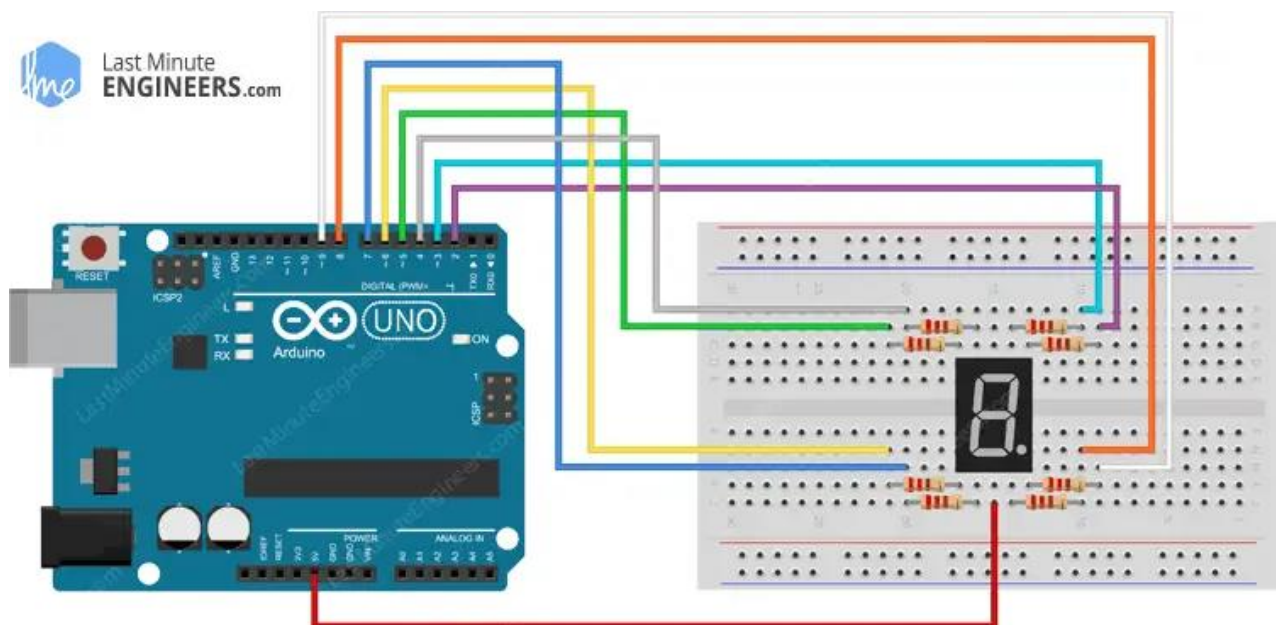
Now connect each segment pin to a digital pin on your Arduino. Connect the four upper pins (b, a, f, and g) to Arduino's digital pins 2 through 5. Then connect the four lower pins (e, d, c, and DP) to Arduino's digital pins 6 through 9.

While the display might work without current-limiting resistors, always include them to protect your display from damage. For this project, you'll need eight  $220\Omega$  resistors – one for each segment. These resistors go between the Arduino's digital pins and the display's segment pins.

Here's a quick reference table for the pin connections:

7-Segment Display	Arduino	Notes
a	3	via $220\Omega$
b	2	via $220\Omega$
c	8	via $220\Omega$
d	7	via $220\Omega$
e	6	via $220\Omega$
f	4	via $220\Omega$
g	5	via $220\Omega$
DP	9	via $220\Omega$
COM	5V	only for common anode display
COM	GND	only for common cathode display

Refer to the figure below to see how everything is connected.

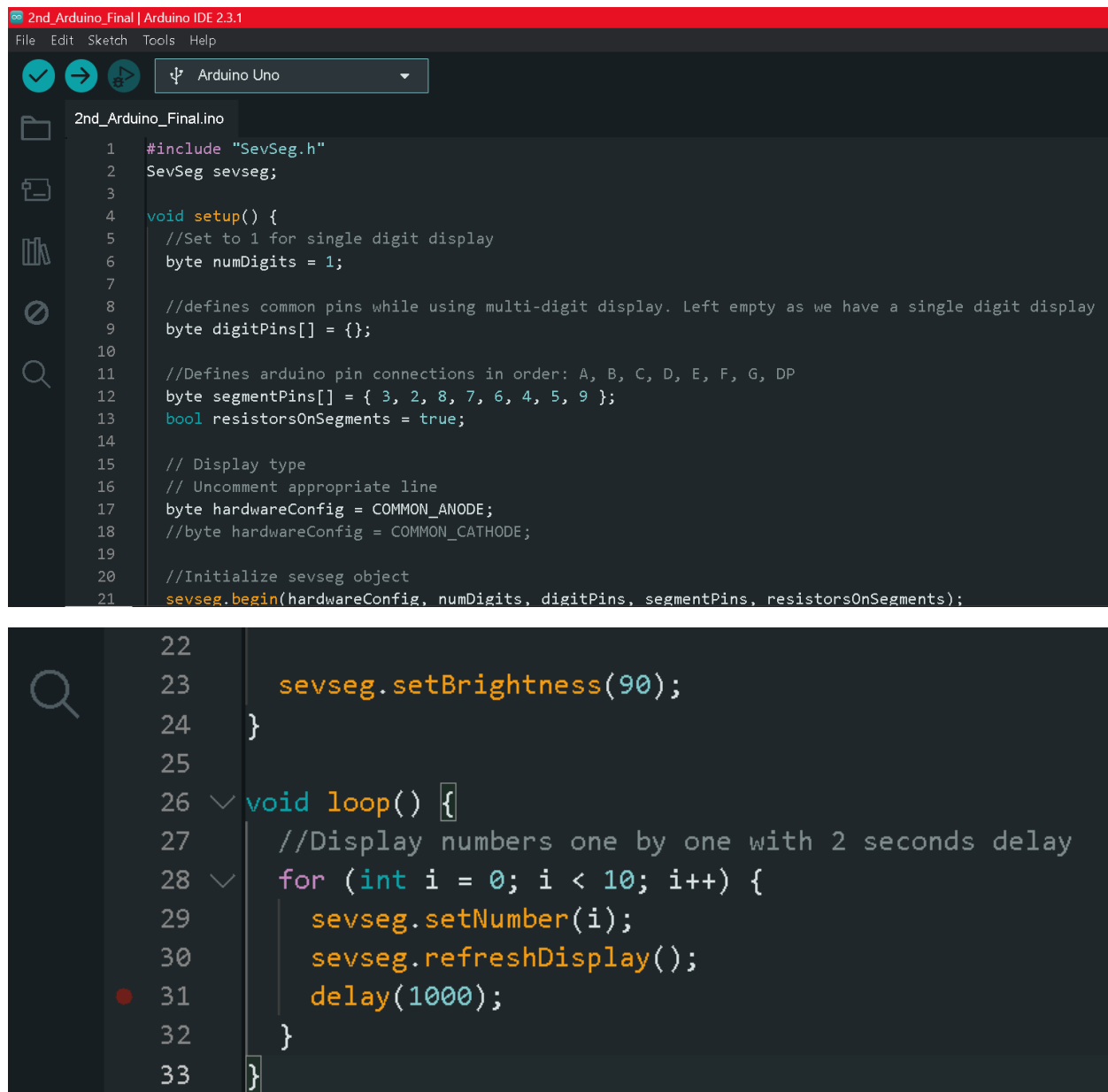


With everything properly connected and protected, you're ready to program your Arduino to display numbers and letters on your 7-segment display!

## CODE

Now let's bring our 7-segment display to life!

The test sketch below will make the display count from 0 to 9. Go ahead and give it a try, and once you've seen it in action, we'll walk through the details together.



```
2nd_Arduino_Final | Arduino IDE 2.3.1
File Edit Sketch Tools Help

2nd_Arduino_Final.ino
1  #include "SevSeg.h"
2  SevSeg sevseg;
3
4  void setup() {
5      //Set to 1 for single digit display
6      byte numDigits = 1;
7
8      //defines common pins while using multi-digit display. Left empty as we have a single digit display
9      byte digitPins[] = {};
10
11     //Defines arduino pin connections in order: A, B, C, D, E, F, G, DP
12     byte segmentPins[] = { 3, 2, 8, 7, 6, 4, 5, 9 };
13     bool resistorsOnSegments = true;
14
15     // Display type
16     // Uncomment appropriate line
17     byte hardwareConfig = COMMON_ANODE;
18     //byte hardwareConfig = COMMON_CATHODE;
19
20     //Initialize sevseg object
21     sevseg.begin(hardwareConfig, numDigits, digitPins, segmentPins, resistorsOnSegments);
22
23     sevseg.setBrightness(90);
24 }
25
26 void loop() {
27     //Display numbers one by one with 2 seconds delay
28     for (int i = 0; i < 10; i++) {
29         sevseg.setNumber(i);
30         sevseg.refreshDisplay();
31         delay(1000);
32     }
33 }
```

**REFERENCE:**

<https://lastminuteengineers.com/seven-segment-arduino-tutorial/>