

Lab 2: SEVEN SEGMENTS

1. Objectives: After this lab, the user will be able to:

- Implement and program 7 segment led

2. Facilities

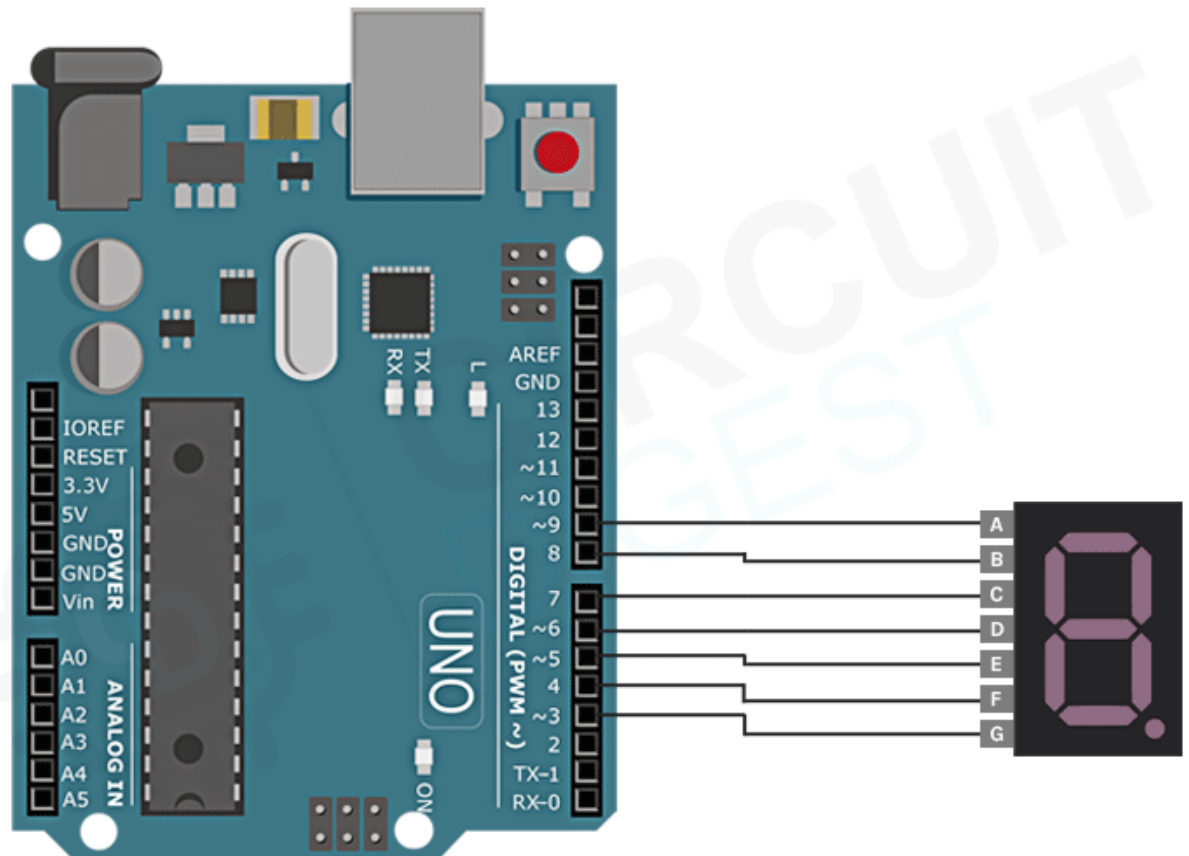
- Proteus software
- Arduino IDE

3. Prerequisite

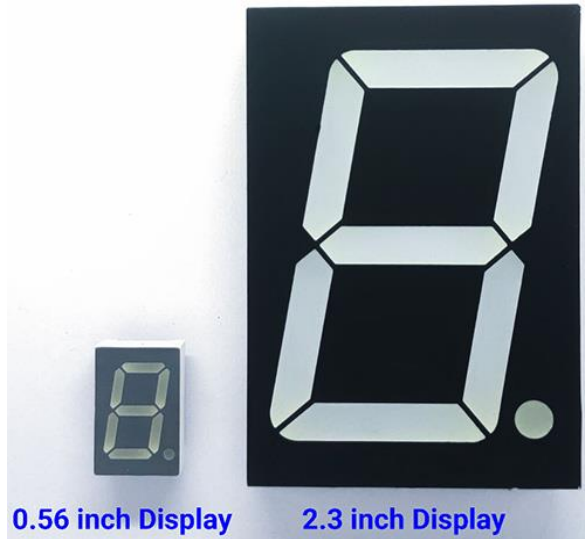
- Basic electronic
- C programming

4. Introduction to 7 segment led

The seven-segment displays are designed for displaying numeric values. You can find them anywhere from instruments to space shuttles. They are the most practical way to display numeric values. They are cheap and easy to use. Not only that, they are highly readable in any light condition, unlike LCDs. You can find them in many everyday usages like counters or token systems, etc.

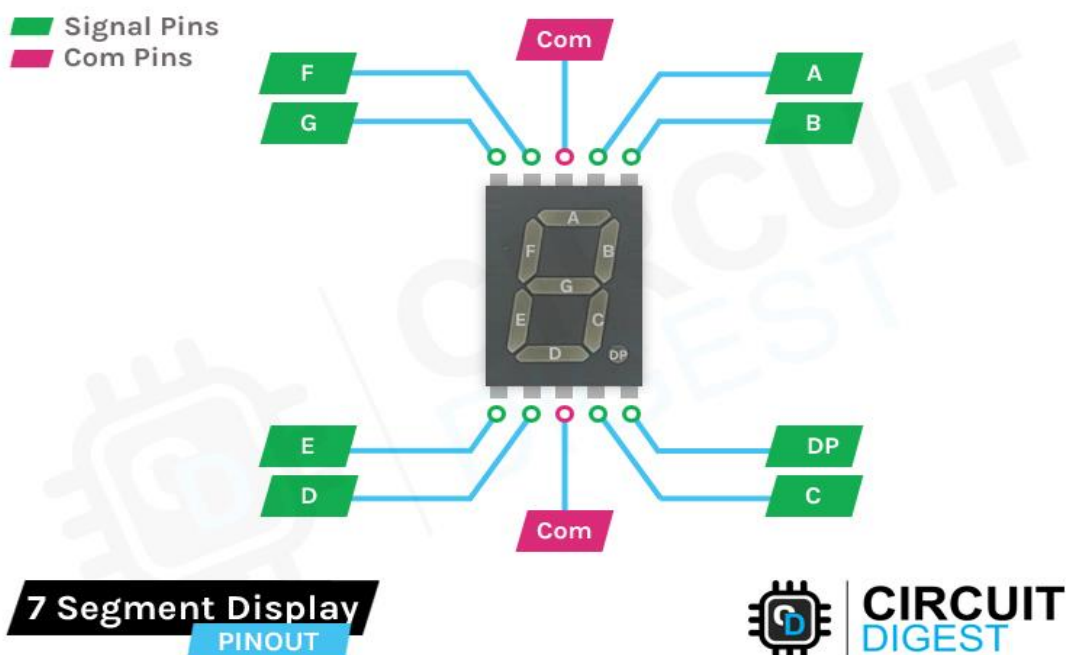


And here is a small simulation showing the Seven Segment Interface. The Arduino will count from 0 to 9 and repeat. The value will be displayed on the Seven Segment display. Seven-segment displays are available in various sizes and colours. They are available from 0.28 inches to 18 inches, and even bigger sizes are available for industrial usage. The **most commonly used display size is 0.56 inches**.



4.1. Seven Segment Display Pinout

As the name suggests, the seven-segment display consists of seven LEDs, which are arranged in a particular order to form the digit eight. And most common modules will come with an additional LED to indicate the decimal point. This will be useful when using multiple display modules to display a decimal number. Each one of these LEDs are arranged in a specific order as a segment and one of its pins is being brought out of the plastic package. The other LED pins are connected together and wired to form a common pin.



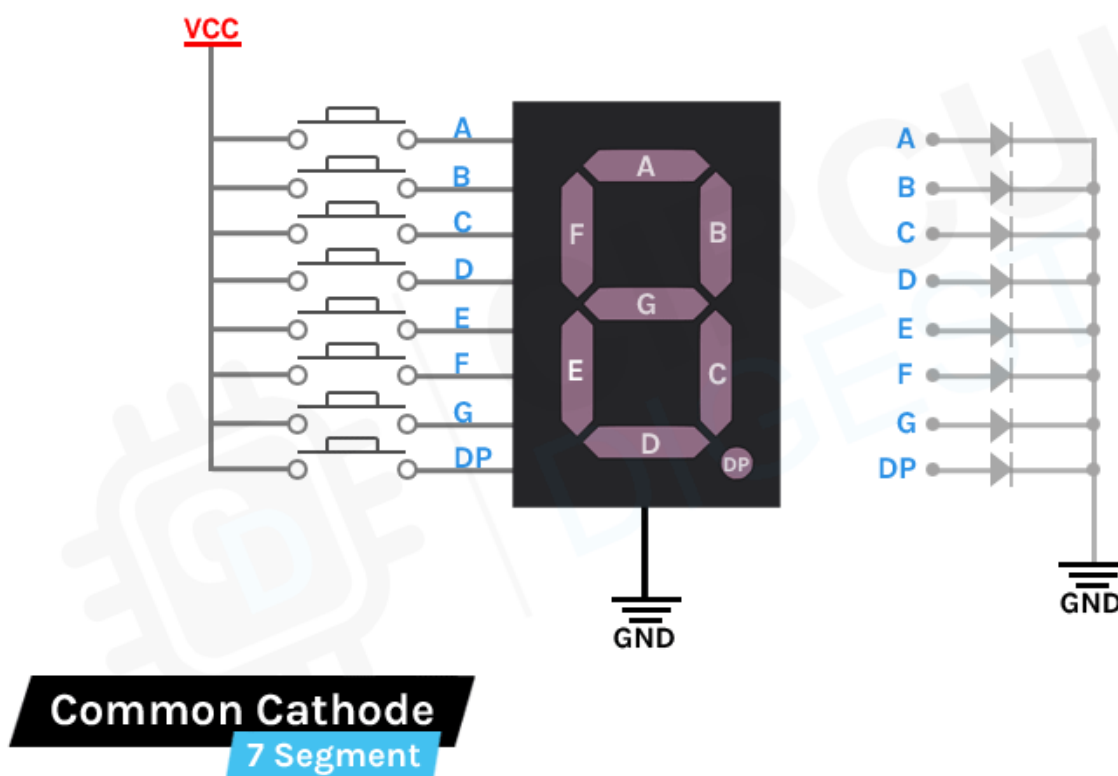
By forward biasing the appropriate pins of LED segments in a particular order, we can generate the character pattern of a desired number on the display. This allows us to display the numbers from 0 to 9 on the display. In the above image, we have highlighted the eight data pins in green and the common pin in pink colours.

4.2. Types of Seven Segment Displays

Seven segment displays are mainly categorized into two types, **Common Cathode (CC)** and **Common Anode (CA)**. This classification is done based on which pin of the LEDs is connected to the common pin. If the Anodes of these LEDs are connected together, then it's called **Common Anode displays**. And if the Cathode is connected together, then they are called **Common Cathode displays**.

Common Cathode Displays

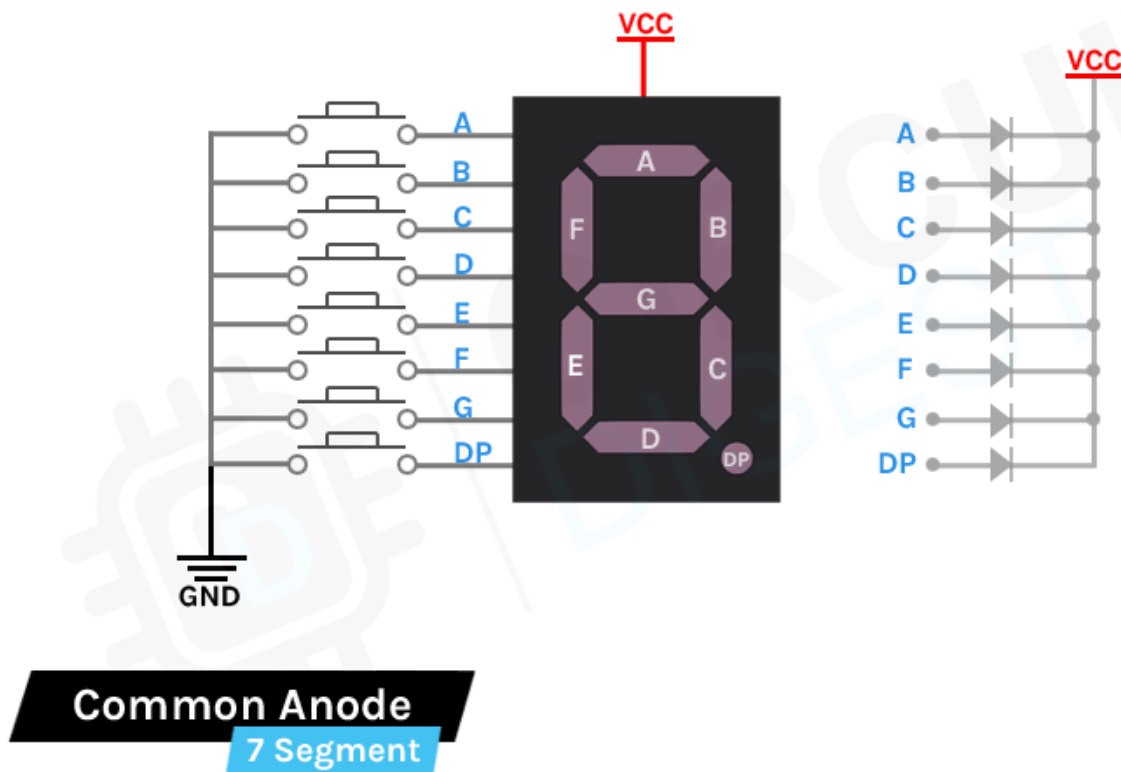
In a Common Cathode (CC) display, all the cathodes of the LEDs are connected together to the common pin. This common pin is then connected to the ground rail of the circuit. The individual LEDs can be activated by applying a high pulse or a logic 1 signal to the corresponding pin.



In the above image, you can see the basic working of the seven-segment. The common pin is connected to the GND and each data pin is connected to VCC through a switch. When a switch is pressed, the corresponding LED will be forward biased, and the segment will light up. On the right side, you can see the simplified diagram of a seven-segment display.

Common Anode Displays

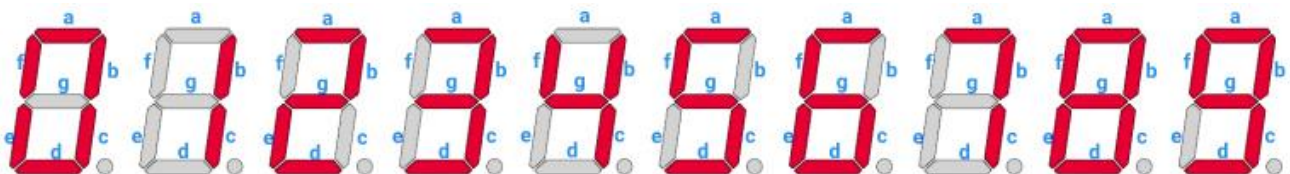
In a Common Anode (CA) display, all the anodes of the LEDs are connected together to the common pin. This common pin is then connected to the positive rail of the circuit. The individual LEDs can be activated by applying a low pulse or a logic 0 signal to the corresponding pin.



In the above image, you can see the basic working of the seven-segment. The common pin is connected to the VCC and each data pin is connected to GND through a switch. When a switch is pressed, the corresponding LED will be forward biased, and the segment will light up. On the right side, you can see the simplified diagram of a seven-segment display.

Displaying Numbers on Seven Segment Display

To display a number or pattern, we must activate the corresponding pins with the appropriate signal, i.e. for Common Cathode display Logic 0 or LOW and for Common Anode display Logic 1 or HIGH. By this method, we can display various digits from 0 to 9. Here is the character map for each character.



And here is the truth table for the Common Cathode display. For the Common Anode display, the signals will be inverted.

A	B	C	D	E	F	G	Display
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

Even though the seven-segments are designed to display numbers, we can also display alphabets using them. Here is the truth table for the alphabet.

A	B	C	D	E	F	G	Display
1	1	1	0	1	1	1	A
0	0	1	1	1	1	1	B
1	0	0	1	1	1	0	C
0	1	1	1	1	0	1	d
1	0	0	1	1	1	1	E
1	0	0	0	1	1	1	F
1	0	1	1	1	1	0	G
0	0	1	0	1	1	1	H
0	0	0	0	1	1	0	I
0	1	1	1	1	0	0	J
1	0	1	0	1	1	1	K
0	0	0	1	1	1	0	L
1	0	1	0	1	0	0	M
1	1	1	0	1	1	0	N
1	1	1	1	1	1	0	O
1	1	0	0	1	1	1	P
1	1	1	0	0	1	1	q
1	1	0	0	1	1	0	R
1	0	1	1	0	1	1	S
0	0	0	1	1	1	1	T
0	1	1	1	1	1	0	U
0	1	1	1	0	1	0	V
0	1	0	1	0	1	0	W
0	1	1	0	1	1	1	X
0	1	1	1	0	1	1	Y
1	1	0	1	0	0	1	Z

5. Practices

5.1. 7-Segment Display with Arduino in Proteus

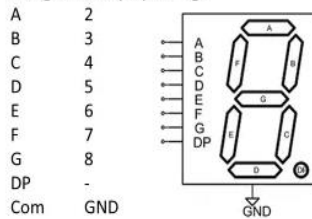
we will learn about interfacing a **7-Segment Display with Arduino Uno** within the Proteus simulation environment. The objective is to **display numbers from 0 to 9**

Connection required:

The seven-segment display will be controlled directly by the Arduino Uno through wiring. You must connect resistors between the display and the Arduino UNO board. The control signals will depend on which number or alphabet needs to be displayed.

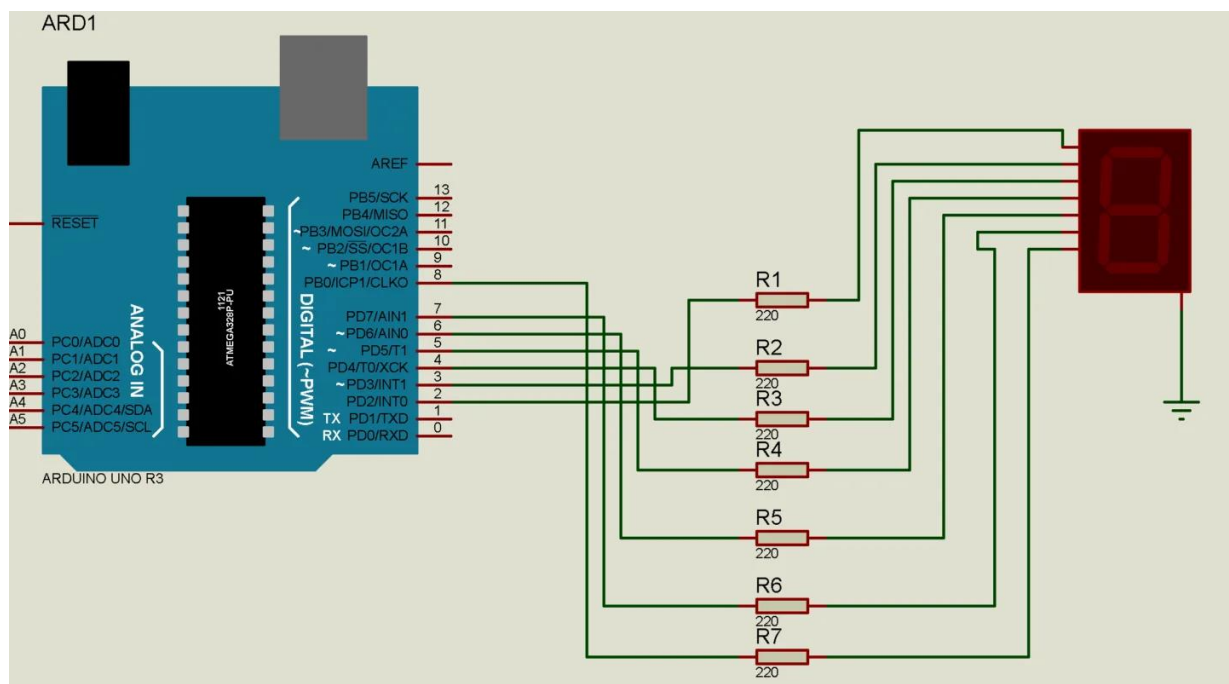
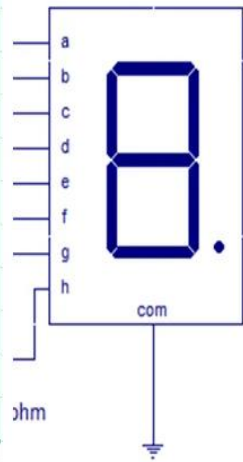
Note: We have used a common anode display in this example; hence, the common pin is connected to 5V. If you're using a common cathode display, connect the common pin to ground

7-Segment Display wiring:



Common Cathode												
Decimal digit	A	B	C	D	a	b	c	d	e	f	g	Display
0	0	0	0	0	1	1	1	1	1	1	0	0
1	0	0	0	1	0	1	1	0	0	0	0	1
2	0	0	1	0	1	1	0	1	1	0	1	2
3	0	0	1	1	1	1	1	1	0	0	1	3
4	0	1	0	0	0	1	1	0	0	1	1	4
5	0	1	0	1	1	0	1	1	0	1	1	5
6	0	1	1	0	1	0	1	1	1	1	1	6
7	0	1	1	1	1	1	1	0	0	0	0	7
8	1	0	0	0	1	1	1	1	1	1	1	8
9	1	0	0	1	1	1	1	1	0	1	1	9

Common cathode seven segment LED display



Arduino Sketch

Now, let's take a look at the code that makes this all work. This Arduino sketch demonstrates the display of digits 0 – 9 and the decimal point.

```
// One digit 7 segment LED display demo. Displays digit 0 - 9
// and decimal point. Prepare array of 7-Arduino pins which we
// need to interfaced with 7segments//
```

```
int segPins[]={2, 3, 4, 5, 6, 7, 8, 9, 0,}; //{a b c d e f g .}
```

```

//Truth table for driving the 7-Segment Display
byte segCode[11][8] = {
  //  a  b  c  d  e  f  g  .
  { 1, 1, 1, 1, 1, 1, 0, 1}, // 0
  { 0, 1, 1, 0, 0, 0, 0, 0}, // 1
  { 1, 1, 0, 1, 1, 0, 1, 0}, // 2
  { 1, 1, 1, 1, 0, 0, 1, 0}, // 3
  { 0, 1, 1, 0, 0, 1, 1, 0}, // 4
  { 1, 0, 1, 1, 0, 1, 1, 0}, // 5
  { 1, 0, 1, 1, 1, 1, 1, 0}, // 6
  { 1, 1, 1, 0, 0, 0, 0, 0}, // 7
  { 1, 1, 1, 1, 1, 1, 1, 0}, // 8
  { 1, 1, 1, 1, 0, 1, 1, 0}, // 9
  { 0, 0, 0, 0, 0, 0, 0, 1}  // .
};

void displayDigit(int digit)
{
  for (int i = 0; i < 8; i++)
  {
    digitalWrite(segPins[i], segCode[digit][i]); //passing the
value pin array
  }
}

void setup()
{
  for (int i = 0; i < 8; i++)
  {
    pinMode(segPins[i], OUTPUT); // declare Arduino pin as an
output
  }
}

void loop()
{
  for (int n = 0; n < 11; n++) // display digits 0 - 9 and
decimal point
  {
    displayDigit(n);
    delay(1000); ///1 second delay
  }
}

```

How the code Works?

Array Declarations: `int segPins[]`: This array holds the pin numbers that are connected to the segments (a, b, c, d, e, f, g, and the decimal point) of the seven-segment display.

`byte segCode[11][8]`: This is a 2D array representing a truth table. It defines the configuration of the individual segments to display each digit (0 – 9) and the decimal point.

displayDigit Function: This function is responsible for displaying a specific digit on the seven-segment display. It takes an argument `digit` (0 – 9) which indicates the digit to be displayed. Inside the function, there's a `for` loop that iterates through the segments (a, b, c, d, e, f, g, and the decimal point). For each segment, it uses `digitalWrite` to set the corresponding pin to the value defined in the `segCode` array. This value determines whether the segment should be lit or not for the given digit.

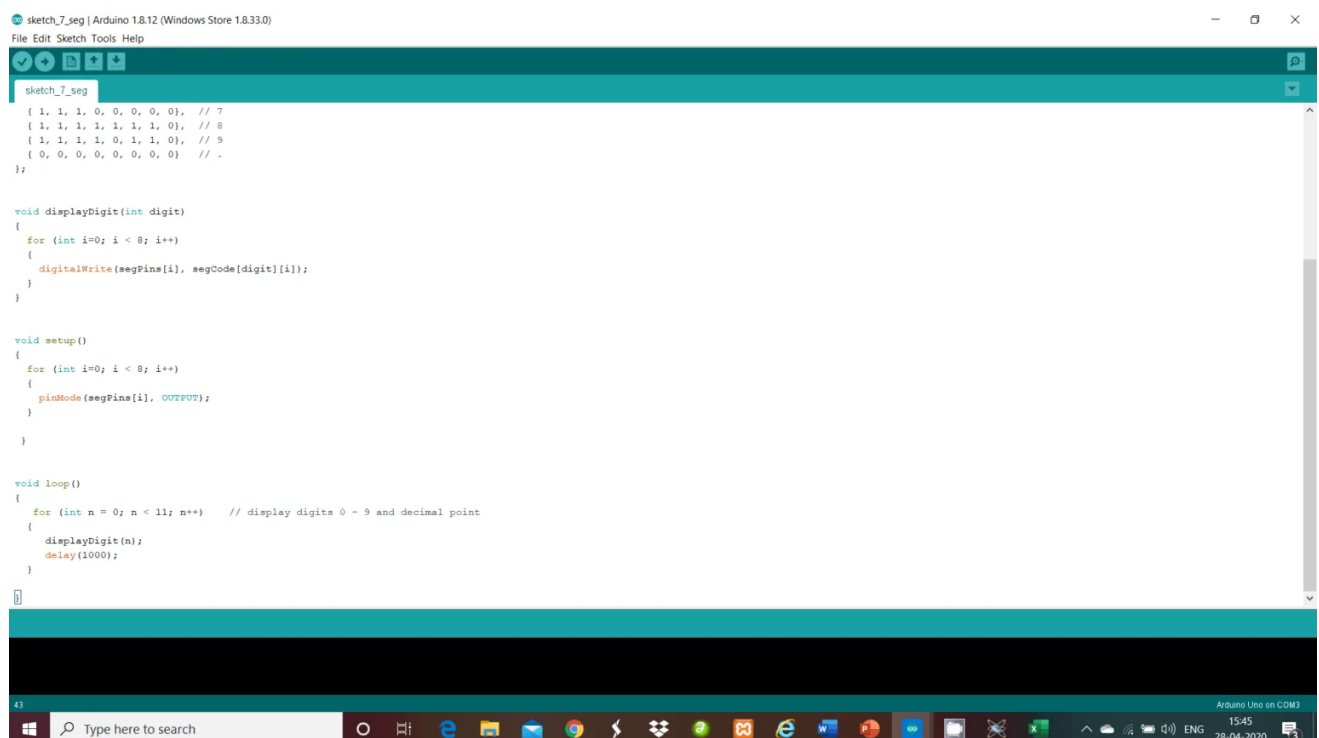
setup Function: In this function, a `for` loop is used to iterate through the `segPins` array.

For each pin in `segPins`, it is configured as an OUTPUT using `pinMode`. This prepares these pins for digital output.

loop Function: The `loop` function is where the main execution takes place. Inside the loop, there's another `for` loop that iterates from `n = 0` to 10. For each value of `n`, the `displayDigit` function is called with `n` as an argument. This 7-Segment Display the digit corresponding to the value of `n` on the seven-segment display. After displaying a digit, there's a `delay(1000)` statement which causes a 1-second delay. This makes each digit stay on the display for 1 second before moving to the next one.

Result:

When you run this code on the Arduino Uno, it continuously cycles through the digits 0 – 9 and the decimal point. The `displayDigit` function configures the individual segments of the seven-segment display according to the truth table (`segCode` array), and the delay between digits creates the illusion of a changing display.



```
sketch_7_seg | Arduino 1.8.12 (Windows Store 1.8.33.0)
File Edit Sketch Tools Help

sketch_7_seg
{ 1, 1, 1, 0, 0, 0, 0, 0}, // 7
{ 1, 1, 1, 1, 1, 1, 0}, // 8
{ 1, 1, 1, 1, 0, 1, 1, 0}, // 9
{ 0, 0, 0, 0, 0, 0, 0, 0} // .
};

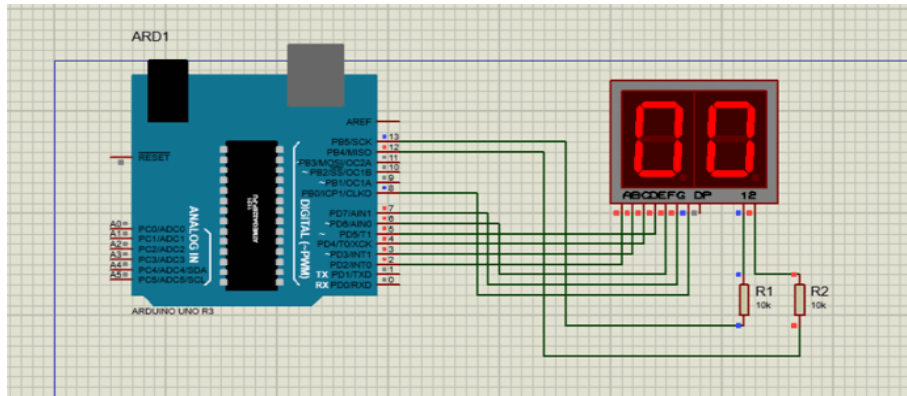
void displayDigit(int digit)
{
  for (int i=0; i < 8; i++)
  {
    digitalWrite(segPins[i], segCode[digit][i]);
  }
}

void setup()
{
  for (int i=0; i < 8; i++)
  {
    pinMode(segPins[i], OUTPUT);
  }
}

void loop()
{
  for (int n = 0; n < 11; n++) // display digits 0 - 9 and decimal point
  {
    displayDigit(n);
    delay(1000);
  }
}
```

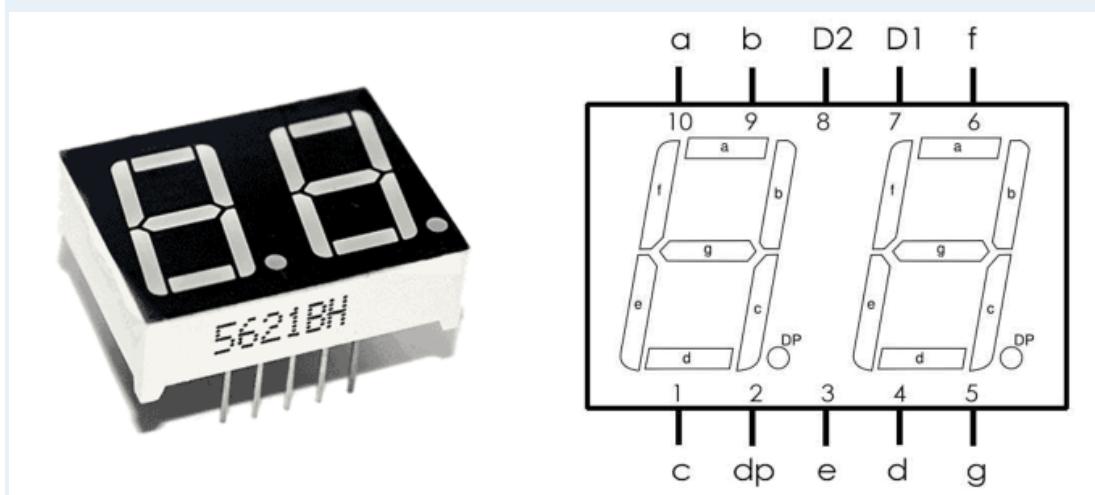

5.2.2-Digit 7 Segment Counter(00-99)with Arduino in Proteus

As we have seen in previous part, for interfacing a single-digit display, we need to connect the common anode pin to +5V supply or we need to connect the common cathode pin to ground, but in case of two digits display we have to drive them independently if we want to display two digits



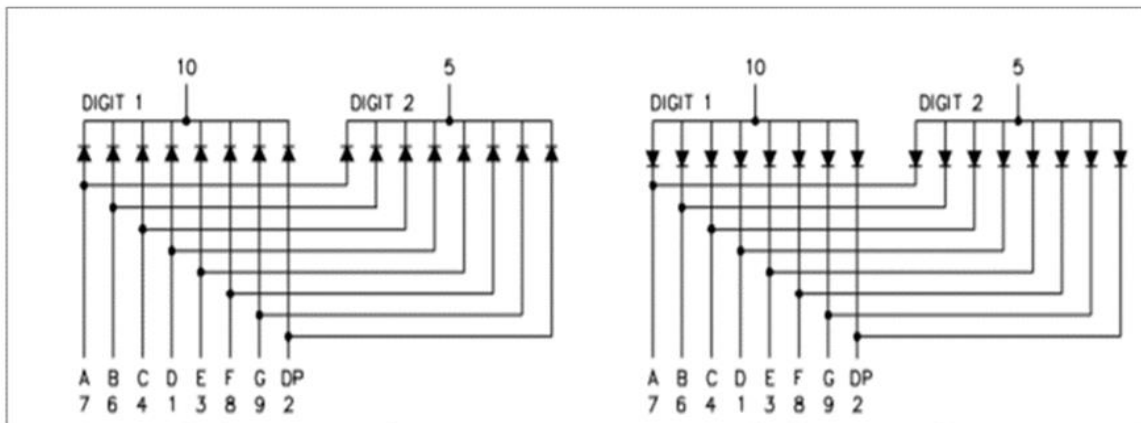
In this part, we use function of the 2-digits. to drive 2 digit 7-segment display we need to used two Arduino I/O pins, each driving a digit of the display individually. This setups is required to drive the common anode or cathode pin using Arduino I/O pins, so they can source enough current to light all seven segments.

Before starting Arduino coding for the project, let us know in brief about 2-Digit 7 Segment displays. A 7-segment display is basically just a couple of regular LEDs behind a block. Each led lights up a particular segment and by lighting a specific combination of LEDs you can represent a number or some letters. Most 7-Segment display are common Cathode, which mean that each LED GND pins (Cathode) are connected together and the VCC+ pins (Anode).

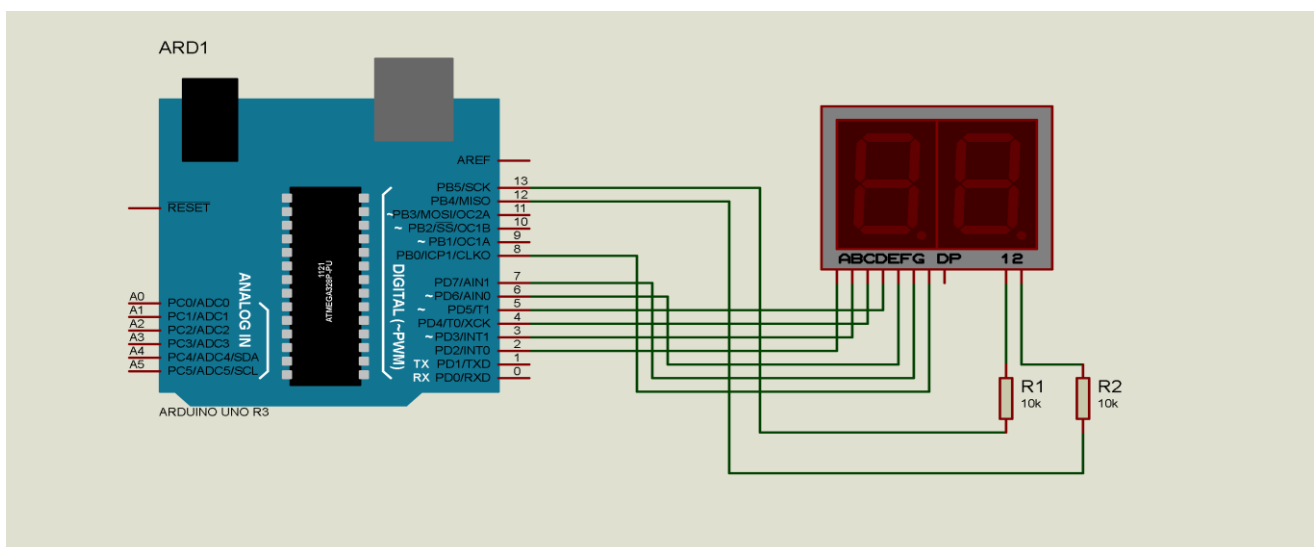


Here's a diagram of the 2-digit 7 Segment display, as you can see pin 1-5 and pin 6, 9 and 10 are connected to a specific segment. Pin 7 and 8 (D1 and D2) are the common Cathode for each digit. So, by grounding D1 or D2 you select which digit you want to light up a specific segment. Of course, like any LED you need to use a resistor(10k) to limit the amount of current drawn by the LED.

Here, 2-Digit 7 Segment is driven directly by Arduino through the wire. Resistors need to be connected between the display and the Arduino UNO board. Depending on which number or alphabet is to be displayed, control signals are applied.



Following schematic shows the 2-Digit 7 Segment with 10-pins that how to set each pin on this LED. a, b, c, d, e, f and g are not on order, so please carefully while doing a wiring. Otherwise your LED display will not show the number you want. “d1” is the power pin to support digit 1 on the right side and “d2” is the power pin to support digit 2 on the left side. “dp” is point pin next to the digit bottom.



Arduino Sketch

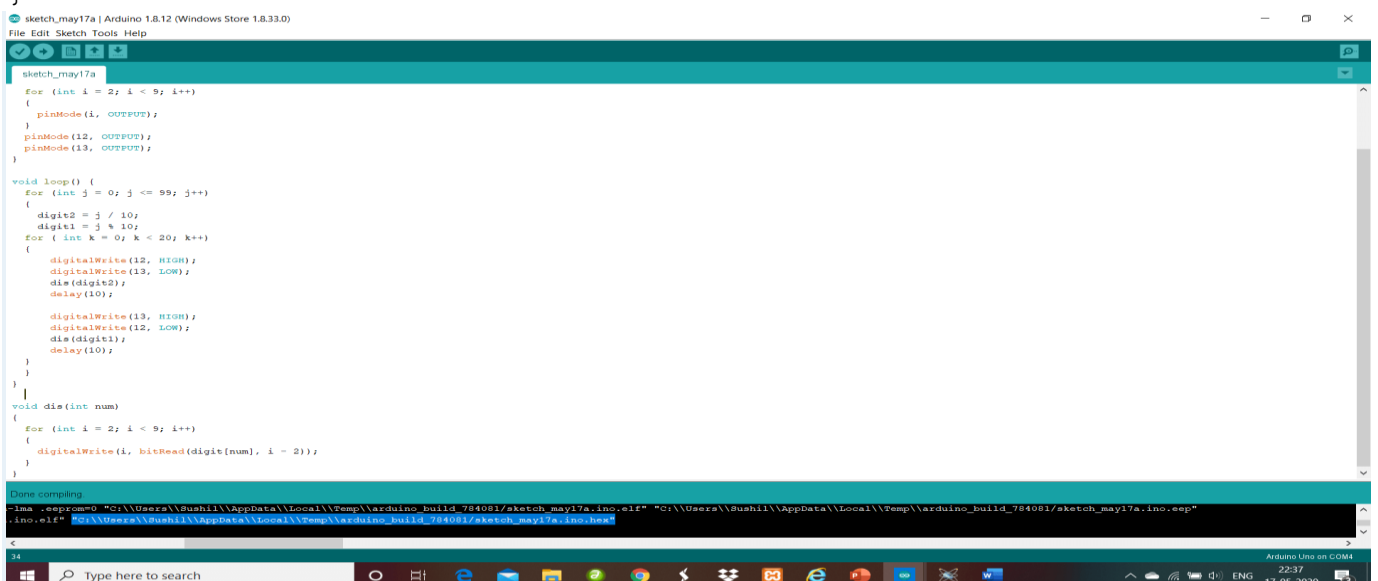
```
//Prepare binary array for all 7 segment to turn on 7 segment
at position of a,b,c,d,e,f,g
int digit[10] = {0b0111111, 0b0000110, 0b1011011, 0b1001111,
0b1100110, 0b1101101, 0b1111101, 0b0000111, 0b1111111,
0b1101111};

int digit1, digit2; // initialize individual digit to controll
each segment
void setup()
{
```

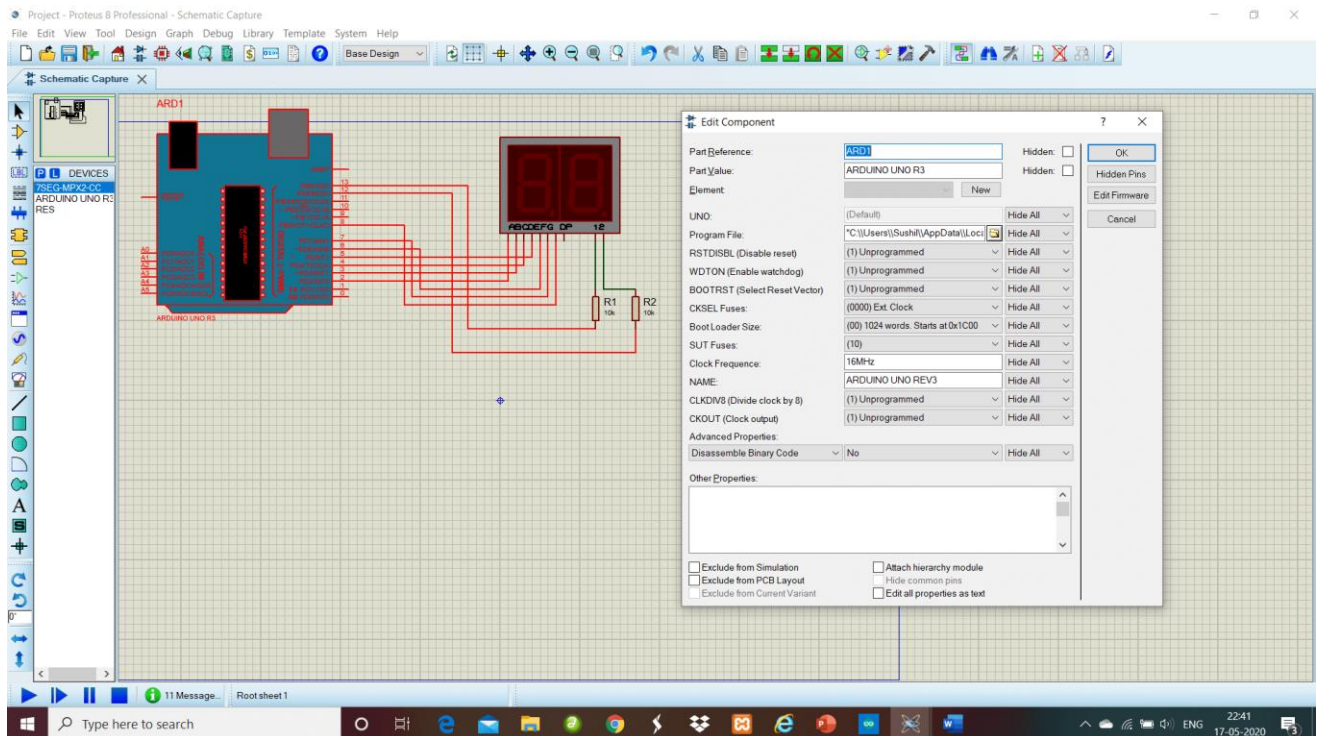
```

for (int i = 2; i < 9; i++)
{
    pinMode(i, OUTPUT); // declare 0-9 th pin as output
}
pinMode(12, OUTPUT); //declare 7 seg Digit1 pin as output
pinMode(13, OUTPUT); //declare 7 seg Digit2 pin as output
}
void loop() {
for (int j = 0; j <= 99; j++) // for loop to pass value from
00-99
{
    digit2 = j / 10;
    digit1 = j % 10;
    for (int k = 0; k < 20; k++) // For loop to control the
digit control to print 00-99
    {
        digitalWrite(12, HIGH);
        digitalWrite(13, LOW);
        dis(digit2);
        delay(10);
        digitalWrite(13, HIGH);
        digitalWrite(12, LOW);
        dis(digit1);
        delay(10);
    }
}
}
void dis(int num)
{
    for (int i = 2; i < 9; i++)
    {
        digitalWrite(i, bitRead(digit[num], i - 2));
    }
}
}

```

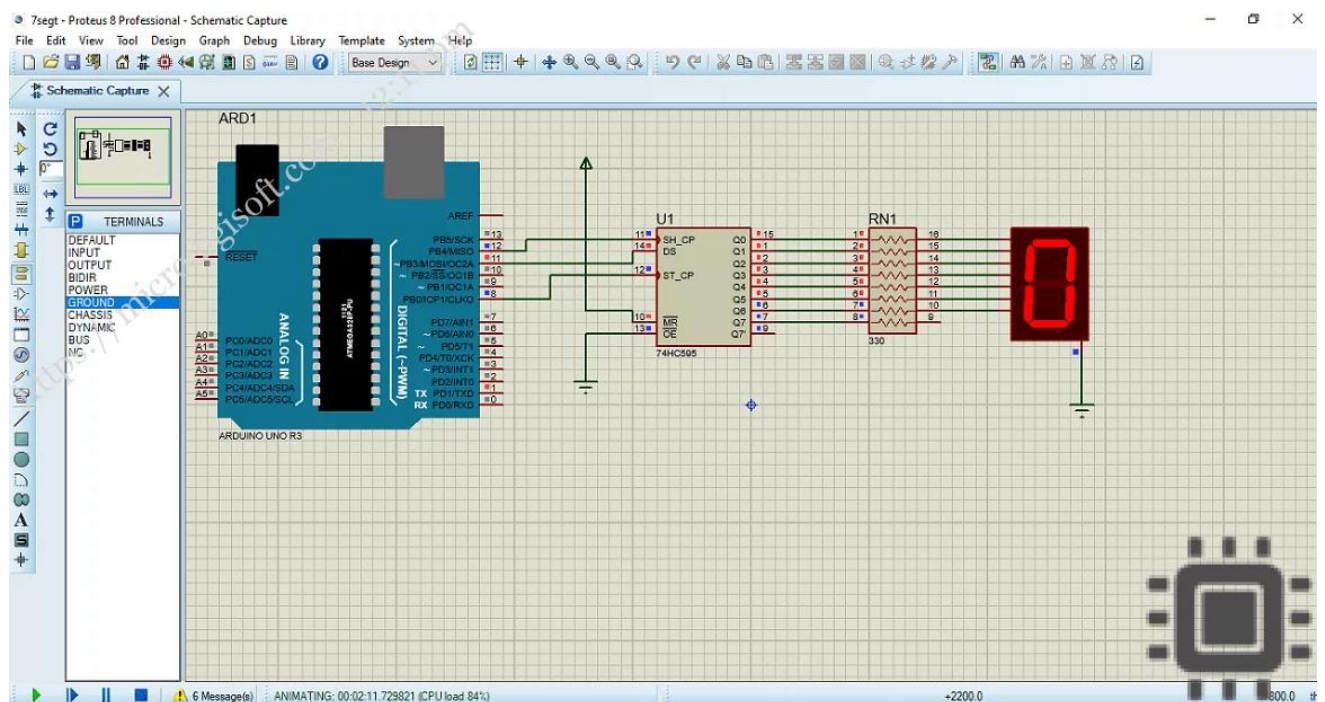


Then get the hex file from the code



5.3. How to Interface 74HC595 IC with 7- Segment & Arduino

Now we discussed about [how to interface 74HC595 IC \(shift Register\) with Arduino board](#), now we will see how to interface [74HC595\(datasheet\)](#) with a **7-segment display and Arduino**. By using a [74HC595 shift register](#) to drive 7-segment displays with Arduino.



In previous section on the [7-segment display interfacing with Arduino](#) in proteus, we have seen that if we interface one 7-segment device directly with Arduino, we will need 8 digital pins of Arduino. Similarly, if we use two-digit, three-digit, four-digit seven-segment

displays, we will need more GPIO pins, even if we use the multiplexing techniques to save microcontroller pins.

74HC595 IC

Therefore, by using a **74HC 595 serial shift register**, we can save Arduino digital pins and can use them for other purposes. For instance, if we use this serial shift register IC, we can interface 7-segment with Arduino by using three pins only, instead of using 8 digital pins.

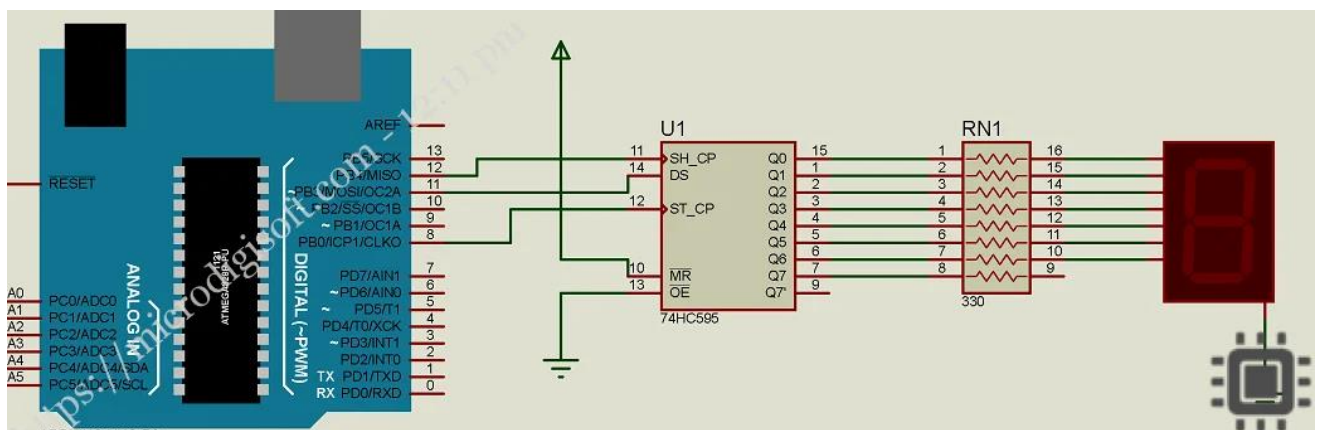
The **7-segment display**, also written as “Seven Segment Display”, consists of seven LEDs (according to their name) arranged in a rectangular fashion as shown. Each of the seven LEDs is called a segment because when illuminated the segment forms part of a numerical digit to be displayed. for more information please visit: [Seven Segment Display with Arduino](#)

Now we can build simple project on seven segment display with Arduino using 75hc595 shift register and Simulated with [Proteus software](#).

Connections:

- Connect Arduino PIN8 to Latch Pin (ST_CP) of the IC.
- Connect Arduino PIN11 to Data Pin (DS) of the IC.
- Connect Arduino Pin12 to Clock Pin (SH_CP) of the IC.
- Connect pin MR of 74HC595 IC to power terminal and OE to ground terminals.
- Connect the output pins (Q1 to Q7) of 74HC595 IC sequentially to seven segment display through a resistor module as shown in following circuit diagram.

If you have a Common Anode Display, connect common to the power terminal, otherwise for a Common Cathode Display connect to the ground terminal (here we use common cathode).



Arduino Sketch

```
#define LATCH_pin 8    // (8) ST_CP [RCK] on 74HC595
#define CLCOK_pin 12   // (12) SH_CP [SCK] on 74HC595
#define DATA_pin 11   // (11) DS [S1] on 74HC595
```

```
unsigned char binary_pattern[] = {
    0b11111100,
```

```

        0b01100000,
        0b11011010,
        0b11110010,
        0b01100110,
        0b10110110,
        0b10111110,
        0b11100000,
        0b11111110,
        0b11100110,
    };
    unsigned int counter=0;
    void clock_signal(void){
        digitalWrite(CLCOK_pin, HIGH);
        delayMicroseconds(500);
        digitalWrite(CLCOK_pin, LOW);
        delayMicroseconds(500);
    }
    void latch_enable(void)
    {
        digitalWrite(LATCH_pin, HIGH);
        delayMicroseconds(500);
        digitalWrite(LATCH_pin, LOW);
    }
    void send_data(unsigned int data_out)
    {
        int i;
        unsigned hold;
        for (i=0 ; i<8 ; i++)
        {
            if ((data_out >> i) & (0x01))
                digitalWrite(DATA_pin,HIGH);
            else
                digitalWrite(DATA_pin,LOW);

            clock_signal();
        }
        latch_enable(); // Data finally submitted
    }

    void setup()
    {
        pinMode(LATCH_pin , OUTPUT);
        pinMode(DATA_pin , OUTPUT);
        pinMode(CLCOK_pin , OUTPUT);
        digitalWrite(LATCH_pin, LOW); // (11) ST_CP [RCK] on 74HC595
        digitalWrite(CLCOK_pin, LOW); // (9) SH_CP [SCK] on 74HC595
        digitalWrite(DATA_pin, LOW); // (12) DS [S1] on 74HC595
        Serial.begin(9600);
    }

```

```
void loop()
{

    send_data(binary_pattern[counter]);
    counter++;
    if(counter>9)
        counter =0;
    delay(1000);
}
```

Then get the **hex file** from the code (At the bottom of screen of the uploading program in Arduino IDE window select the file path). Then double click “**ARDUINO UNO R3**”, we will see Edit Component window after that Browse it in Program File/ Put file path which selecting from the Arduino uploading window, then **click OK**.

5.4.

6.