

Embedded Systems Project Report

Two-Digit Up-Down Counter Using Arduino & 7 Segment Display

CONTENTS:

- Scoreboard Counter using Arduino.
- 00 to 99 up-down counter(with increment and decrement operations) using Arduino.

GROUP MEMBERS:

COE17B015

CED17I045

COE17B014

COE17B046

CED17I027

Scoreboard Counter-using Arduino

The list of required components:

- 1) Arduino UNO
- 2) 1K Resistors - Qty. 6 (10K can also be used)
- 3) 7 Segment Displays (Common Cathode) - Qty. 2
- 4) CD4511 - BCD to 7 Segment Decoder IC - Qty - 2
- 5) Jumper Wires.
- 6) Three Push Buttons- Two buttons for increasing scores of 2 7 segment displays and one for Reset
- 7) Breadboard Qty-1

We can simulate the same using TinkerCad. Link: [Scoreboard Counter](#)

7 Segment Display Pin Configuration

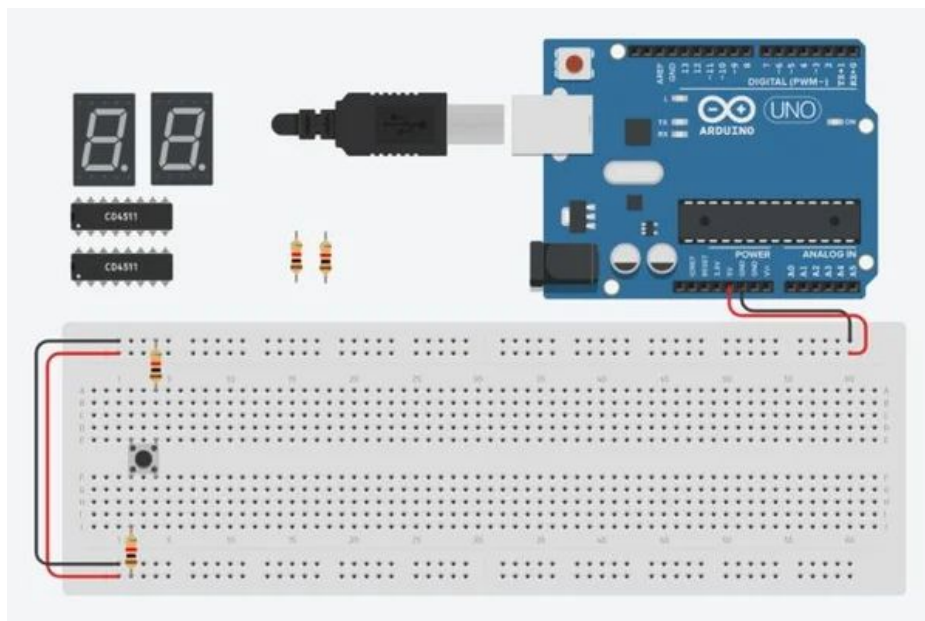
Pin Number	Pin Name	Description
1	e	Controls the left bottom LED of the 7-segment display
2	d	Controls the bottom most LED of the 7-segment display
3	Com	Connected to Ground/Vcc based on type of display
4	c	Controls the right bottom LED of the 7-segment display
5	DP	Controls the decimal point LED of the 7-segment display
6	b	Controls the top right LED of the 7-segment display
7	a	Controls the top most LED of the 7-segment display
8	Com	Connected to Ground/Vcc based on type of display
9	f	Controls the top left LED of the 7-segment display
10	g	Controls the middle LED of the 7-segment display

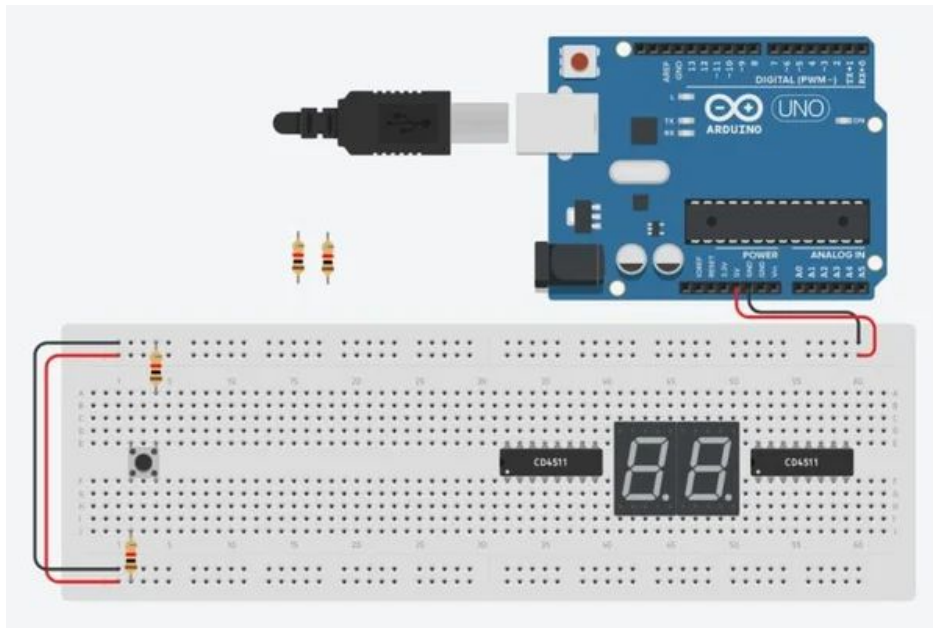
Steps involved

STEP 1: Placing the Components on BreadBoard

We place the components as shown on the breadboard.

Connect one terminal of the push button to 5V through a resistor and connect others to the Ground through the resistor. Similarly connect the other two push buttons.





STEP 2: BCD to 7 Segment Logic

BCD Stands for Binary Coded Decimal, where a decimal number is represented as a 4 Bit Binary.

We are using a BCD to 7 Segment Decoder because it will reduce the number of Arduino Digital Pinouts used to connect single 7 Segment Display. A 7 Segment Display will require a minimum of 7 Arduino pinouts and two of them will require a minimum of 14 while we have only 13 Digital Pinouts on Arduino Uno.

A BCD to 7 Segment Decoder will require only 4 Arduino Pinouts and two of them will use only 8 Digital Pinouts.

BCD to Decimal Mapping is as follows.



BCD ---- DECIMAL

0000 ---- 0

0001 ---- 1

0010 ---- 2

0011 ---- 3

0100 ---- 4

0101 ---- 5

0110 ---- 6

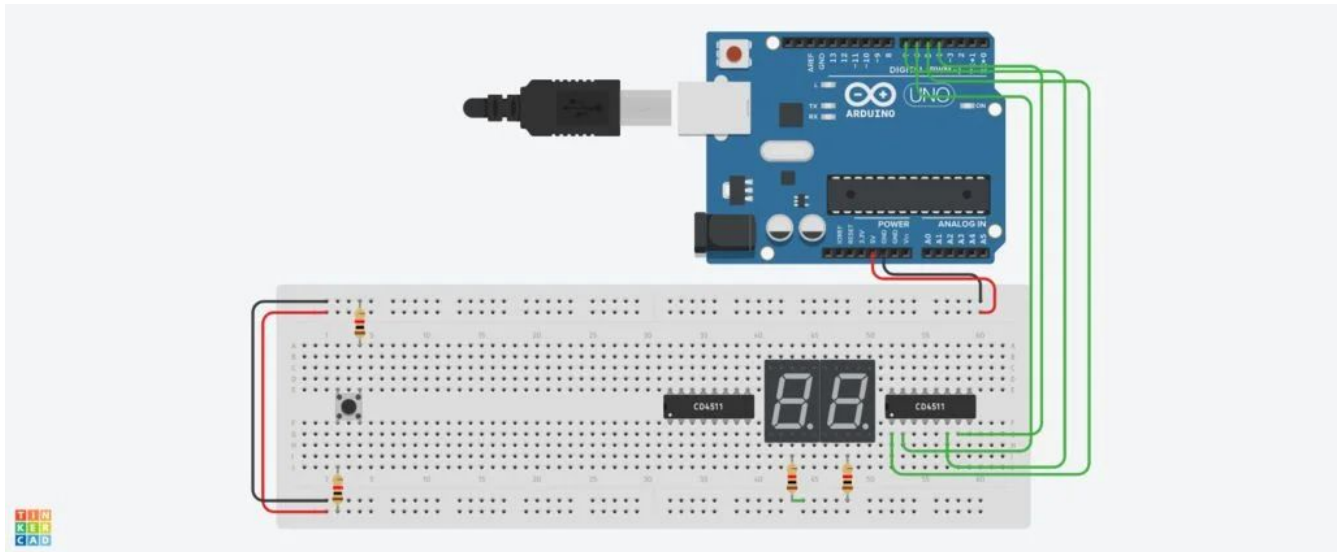
0111 ---- 7

1000 ---- 8

1001 ---- 9

The BCD to Decimal Conversion goes till 15 but we require it till 9 as its the maximum number that can be displayed by single 7 - Segment Display

STEP 3: Connecting Arduino Pinout to BCD Inputs



Here we are using IC CD4511 7 segment Decoder,

The Pins 1,2,6,7 are the BCD input pins of IC CD4511.

Where,

Pin 7 = Bit 0

Pin 1 = Bit 1

Pin 2 = Bit 2

Pin 6 = Bit 3

We'll connect only one of the two ICs with Arduino at first, the one on the right side of the Display

Now for the first decoder IC i.e. the one on the right of Display..

CD4511(1) Arduino

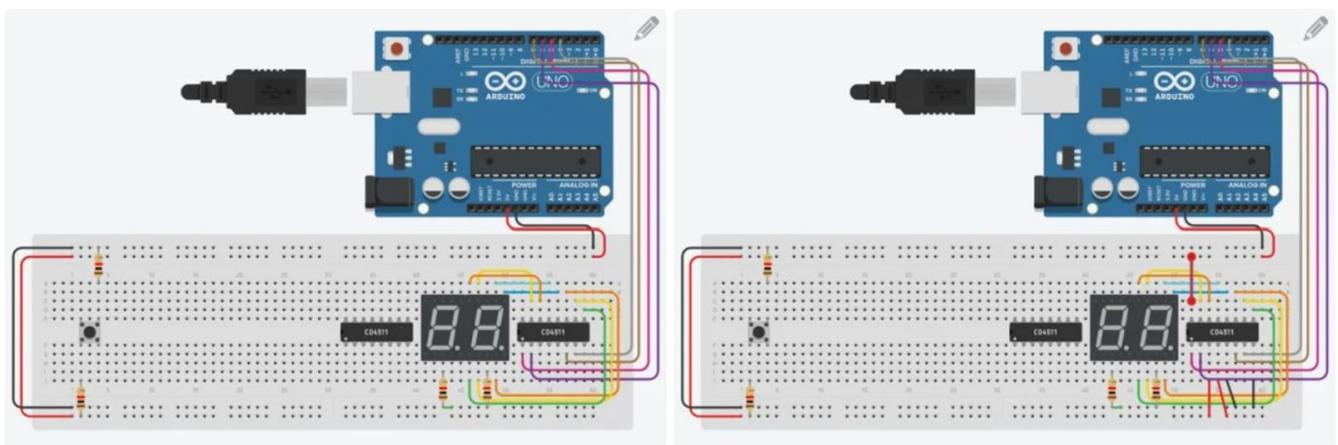
Pin 7 -----> Pin 4

Pin 1 -----> Pin 5

Pin 2 -----> Pin 6

Pin 6 -----> Pin 7

STEP 4: Connecting the 7 Segment Display and Powering the IC



Pins 9 to 15 on the IC are pinouts corresponding to 7 Segment Display.

The mapping of IC pinouts to 7 Segment Display is as follows.

CD4511 ----> 7 Segment Display

13 ----> a

12 ----> b

11 ----> c

10 ----> d

9 ----> e

15 ----> f

14 ----> g

Connect the right-hand side display to Decoder 1 on the left as explained
To Power the IC we need to make following connections for CD4511

CD4511

Pin 3 ---> Vcc (+5v)

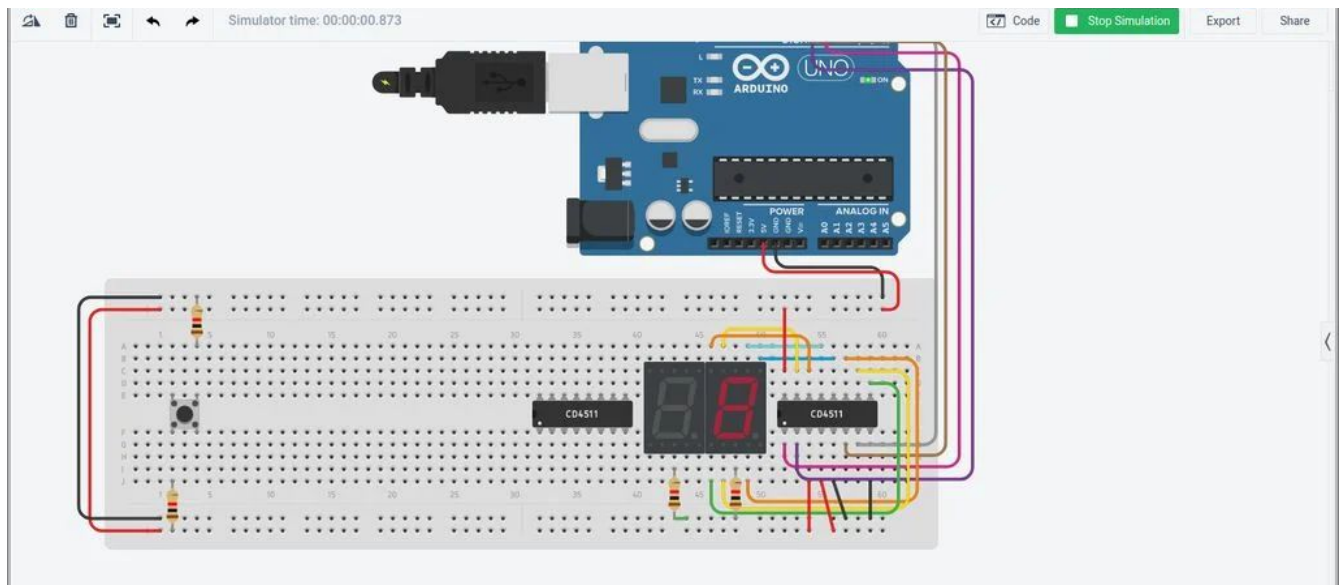
Pin 4 ---> Vcc (+5v)

Pin 5 ---> GND

Pin 8 ---> GND

Pin 16 ---> Vcc(+5V)

STEP 5:Arduino Sketch for Testing First Display

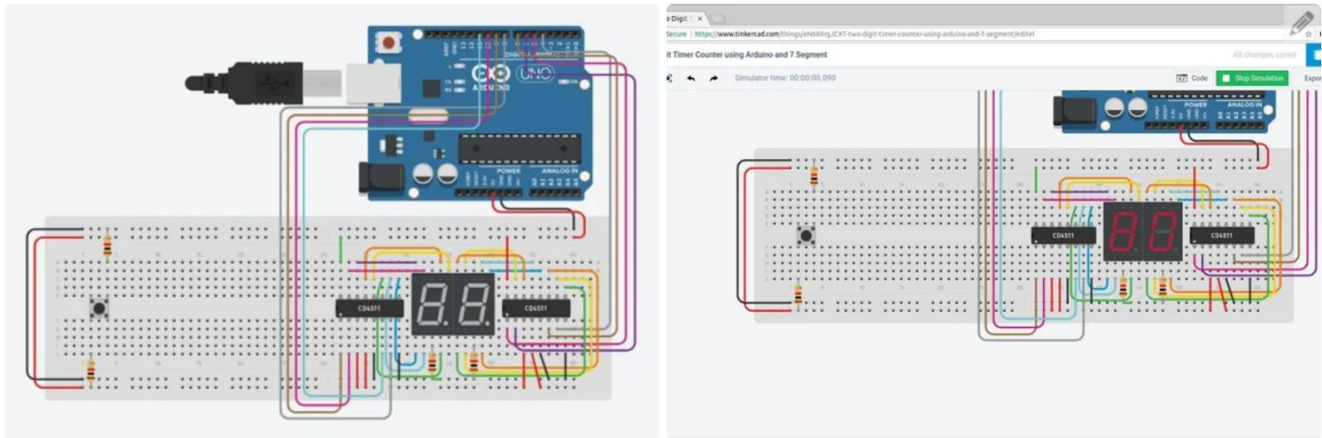


The Arduino Sketch is as follows. Here we are displaying number 8 to check that all the segments are working.

Code for single 7 segment display:

```
int a1 = 4;
int a2 = 5;
int a3 = 6;
int a4 = 7;
void setup()
{
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
}
void loop()
{
  disp1(8);
}
void disp1(int num)
{
  if(num == 8)
  {
    digitalWrite(a1, LOW);
    digitalWrite(a2, LOW);
    digitalWrite(a3, LOW);
    digitalWrite(a4, HIGH);
  }
}
```

Step 6: Connecting the Other Display and Testing It.



The Arduino pinout to CD4511(2), i.e the second decoder, the one on the left, is as follows.

CD4511 (2) -----> Arduino

Pin 7 -----> 8

Pin 1 -----> 9

Pin 2 -----> 10

Pin 6 -----> 11

Connect the 7- Segment Pins of the IC to the other Display as shown in step 4. and make power connections.

```
//BCD 1
```

```
int a1 = 4;
```

```
int a2 = 5;
```

```
int a3 = 6;
```

```
int a4 = 7;
```

```
//BCD 2
int b1 = 8;
int b2 = 9;
int b3 = 10;
int b4 = 11;

void setup()
{
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
  pinMode(8,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(11,OUTPUT);
}

void loop()
{
  disp1(8);
}
void disp1(int num)
{
  if(num == 8)
  {
    digitalWrite(a1, LOW);
    digitalWrite(a2, LOW);
    digitalWrite(a3, LOW);
    digitalWrite(a4, HIGH);
  }
}

void disp2(int num)
{
  if(num == 8)
  {
```

```
digitalWrite(b1, LOW);  
digitalWrite(b2, LOW);  
digitalWrite(b3, LOW);  
digitalWrite(b4, HIGH);  
}  
}
```

STEP 7: Completing the Display Functions for Displaying Numbers 0 to 9.

In the display function disp1(), make the following changes

```
void disp1(int num)  
{  
  if(num==0)//0000  
  {  
    digitalWrite(a1, LOW);  
    digitalWrite(a2, LOW);  
    digitalWrite(a3, LOW);  
    digitalWrite(a4, LOW);  
  }  
  if(num == 1)//0001  
  {  
    digitalWrite(a1, HIGH);  
    digitalWrite(a2, LOW);  
    digitalWrite(a3, LOW);  
    digitalWrite(a4, LOW);  
  }  
  if(num == 2)//0010  
  {  
    digitalWrite(a1, LOW);//0  
    digitalWrite(a2, HIGH);//1  
    digitalWrite(a3, LOW);//0  
    digitalWrite(a4, LOW);//0  
  }  
  if(num == 3)//0011  
  {  
    digitalWrite(a1, HIGH);//1  
    digitalWrite(a2, HIGH);//1
```

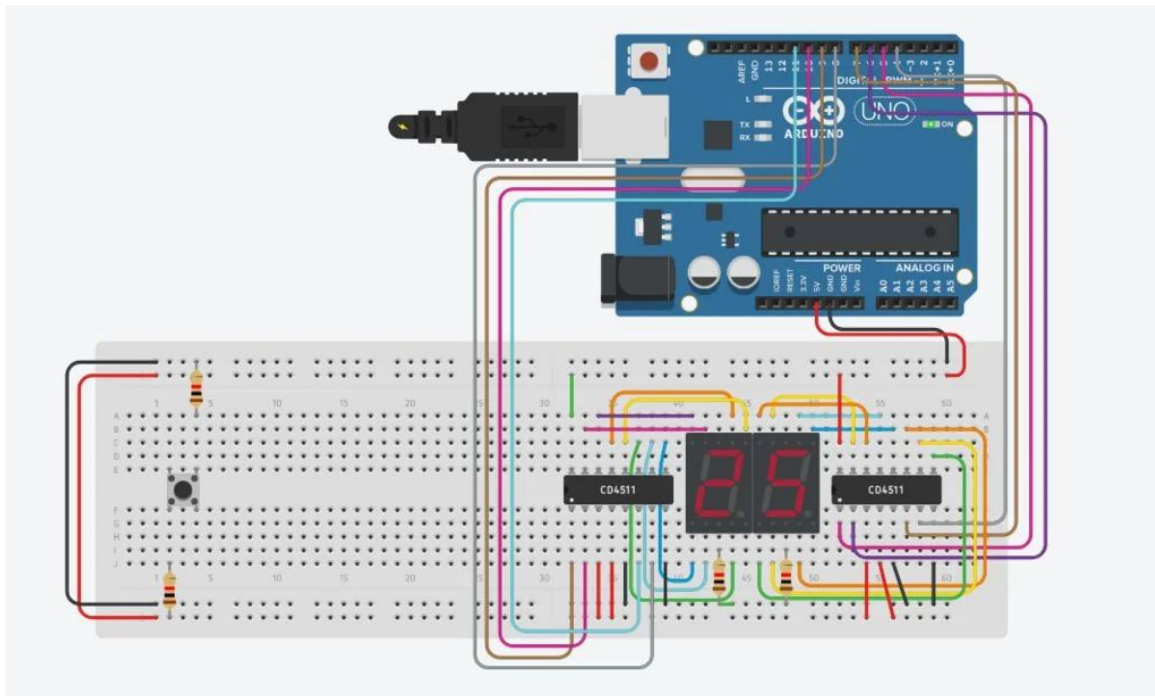
```
digitalWrite(a3, LOW);//0
digitalWrite(a4, LOW);//0
}
if(num == 4)//0100
{
    digitalWrite(a1, LOW);//0
    digitalWrite(a2, LOW);//0
    digitalWrite(a3, HIGH);//1
    digitalWrite(a4, LOW);//0
}
if(num == 5)//0101
{
    digitalWrite(a1, HIGH);//1
    digitalWrite(a2, LOW);//0
    digitalWrite(a3, HIGH);//1
    digitalWrite(a4, LOW);//0
}
if(num == 6)//0110
{
    digitalWrite(a1, LOW);//0
    digitalWrite(a2, HIGH);//1
    digitalWrite(a3, HIGH);//1
    digitalWrite(a4, LOW);//0
}
if(num == 7) //0111
{
    digitalWrite(a1, HIGH);//1
    digitalWrite(a2, HIGH);//1
    digitalWrite(a3, HIGH);//1
    digitalWrite(a4, LOW);//0
}
if(num == 8) //1000
{
    digitalWrite(a1, LOW);//0
    digitalWrite(a2, LOW);//0
    digitalWrite(a3, LOW);//0
    digitalWrite(a4, HIGH);//1
}
if(num == 9)//1001
{
    digitalWrite(a1, HIGH);//1
    digitalWrite(a2, LOW);//0
    digitalWrite(a3, LOW);//0
    digitalWrite(a4, HIGH);//1
}
}
```

Similarly we can write the function for 2nd display

```
void disp2(int num)
{
    if(num==0)
    {
        digitalWrite(b1, LOW);//0
        digitalWrite(b2, LOW);//0
        digitalWrite(b3, LOW);//0
        digitalWrite(b4, LOW);//0
    }
    if(num == 1)//0001
    {
        digitalWrite(b1, HIGH);//1
        digitalWrite(b2, LOW);//0
        digitalWrite(b3, LOW);//0
        digitalWrite(b4, LOW);//0
    }
    if(num == 2)//0010
    {
        digitalWrite(b1, LOW);//0
        digitalWrite(b2, HIGH);//1
        digitalWrite(b3, LOW);//0
        digitalWrite(b4, LOW);//0
    }
    if(num == 3)//0011
    {
        digitalWrite(b1, HIGH);//1
        digitalWrite(b2, HIGH);//1
        digitalWrite(b3, LOW);//0
        digitalWrite(b4, LOW);//0
    }
    if(num == 4)//0100
    {
        digitalWrite(b1, LOW);//0
        digitalWrite(b2, LOW);//0
        digitalWrite(b3, HIGH);//1
        digitalWrite(b4, LOW);//0
    }
    if(num == 5) //0101
```

```
{
    digitalWrite(b1, HIGH); //1
    digitalWrite(b2, LOW); //0
    digitalWrite(b3, HIGH); //1
    digitalWrite(b4, LOW); //0
}
if(num == 6) //0110
{
    digitalWrite(b1, LOW); //0
    digitalWrite(b2, HIGH); //1
    digitalWrite(b3, HIGH); //1
    digitalWrite(b4, LOW); //0
}
if(num == 7) //0111
{
    digitalWrite(b1, HIGH); //1
    digitalWrite(b2, HIGH); //1
    digitalWrite(b3, HIGH); //1
    digitalWrite(b4, LOW); //0
}
if(num == 8) //1000
{
    digitalWrite(b1, LOW); //0
    digitalWrite(b2, LOW); //0
    digitalWrite(b3, LOW); //0
    digitalWrite(b4, HIGH); //1
}
if(num == 9) //1001
{
    digitalWrite(b1, HIGH); //1
    digitalWrite(b2, LOW); //0
    digitalWrite(b3, LOW); //0
    digitalWrite(b4, HIGH); //1
}
}
```


Step 8: Connecting and writing logic for buttons

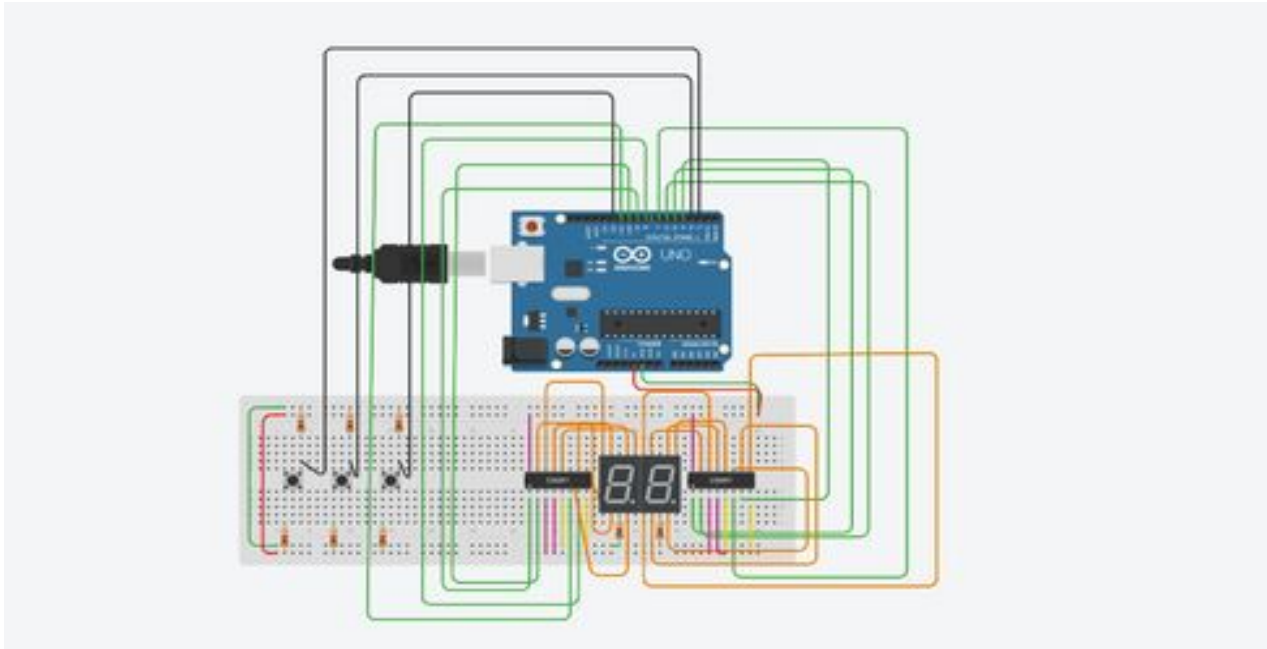


Here, we write the logic for the three buttons.

One button is for increasing the score on 1 display and other for the 2nd display. A third button is to reset the scores to 0 on both buttons. Third button is connected to the 12 th pin of arduino.

After all the connections are done our circuit of scoreboard looks like this

Final Circuit image:



Logic:

```
int number1=0;//first scoreboard
int number2=0;//second scoreboard
int button = 2;// button for first scoreboard
int button1=3;// button for second scoreboard
int reset_button=12;// reset button
int state = 0;//state for first button
int state1=0;//state for second button
int reset_state=0;// state for reset button
```

We then modify the void loop function and write the logic to display the scores on two displays and the reset button logic. We also take care of the overflow conditions where the scores become greater than ten in this case we reset the score to 0 on that button.

Final Code for void loop function.

```
void loop()
{
  state = digitalRead(button);
  state1 = digitalRead(button1);
  reset_state = digitalRead(reset_button);
  if(state == HIGH)
  {
    number1++;
  }
  if(state1 == HIGH)
  {
    number2++;
  }
  if(reset_state==HIGH)
  {
    number1=0;
    number2=0;
  }
  if(number1==10)
  {
    number1=0;
  }
  if(number2==10)
  {
    number2=0;
  }
  disp1(number1);
  disp2(number2);
  delay(100);
}
```

The buttons are assigned as inputs in the setup() function and the following lines are added.

```
pinMode(button,INPUT);
pinMode(button1,INPUT);
pinMode(reset_button,INPUT);
```

Simulation:

Now we can simulate our scoreboard logic.

To simulate


1. First button is for increasing the score on one display
2. Second button is for increasing the score on 2nd display(second from right in the circuit)
3. Third button is to reset the scores on both displays.
4. If a score reaches 10 its reset to 0 on the display.

The simulation video is attached to the parent folder in the drive.

Full Code for the project is attached as a .ino file

Filename : **group9_scoreboardCounter.ino**

Simulation Link of TinkerCad project - [Scoreboard Counter](#)



Two digit- 00 to 99 Counter-using Arduino(with increment and decrement operations)

The list of required components

- 1) Arduino UNO
- 2) 1K Resistors - Qty. 4 (10K can also be used)
- 3) 7 Segment Displays (Common Cathode) - Qty. 2
- 4) CD4511 - BCD to 7 Segment Decoder IC - Qty - 2
- 5) Jumper Wires.
- 6) Two Push Buttons- One for increasing and other for decreasing count.
- 7) Breadboard - Qty 1

We can simulate the same using TinkerCad. Link: [Up down counter](#)

Procedure:

For the connections and circuit we follow the steps used for the scoreboard counter.

In this up-down counter we only change the logic of the circuit and we reduce the number of buttons as only two buttons are required

1. Increasing the number(count)
2. Decreasing the number(count)

Here the disp1 and disp2 functions remain the same and we change the logic in the loop and setup functions.

Steps followed:

1. To display a two-digit number we will write a code to separate the digits of a two-digit number and store their values into d1 and d2 where d1 is the digit at ones place and d2 is the digit at tens place.
2. Which we will then write into two displays using the disp1() and disp(2) functions, where disp1() controls the data to binary inputs of 7 Segment decoder 1 (i.e the one on right) that in turn controls 7 Segment Display at ones place, and disp2() controls the data to binary inputs of 7 Segment decoder 2 (i.e the one on left) that in turn controls 7 Segment Display at tens place.
3. First of all,we declare the following integer variables globally

```
int num=0;
int done;//To store units place
int dten;//To store tens place
```

4. We modify the void loop function in this way

```
void loop()
{
  state = digitalRead(button);
  state1 = digitalRead(button1);
  if(state == HIGH)
  {
    num++;
  }
  if(state1==HIGH)
  {
    num--;
  }
  if(num==100)
  {
    num=0;
  }
  if(num== -1)
  {
    num=99;
  }
  done=num%10;
  dten=num/10;
  disp1(done);
  disp2(dten);
  delay(100);
}
```

If we take an example of 34 we have to display 34

done = 34%10 stores 4 which is units place and 4 is displayed using disp1 function

dten = 34/10 stores 3 which is tens place and 3 is displayed using disp2 function

5. Now we connect the buttons and one button is for displaying the digit in units place and other in tens place and setup and loop functions are modified accordingly.

Changes in void loop()

```
state = digitalRead(button);
```

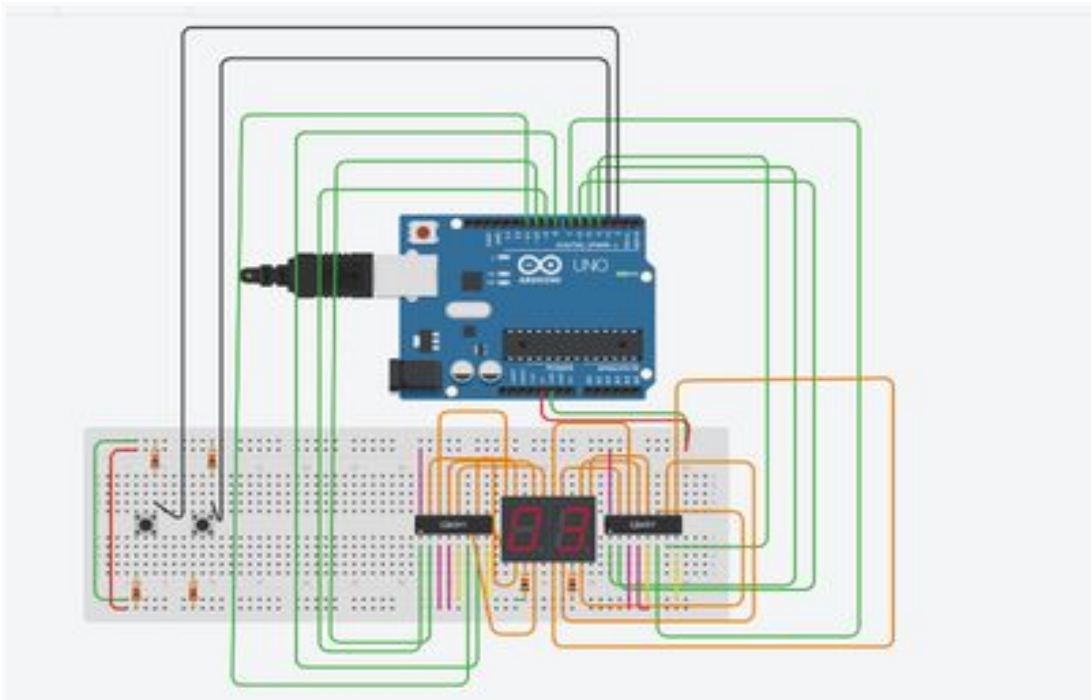
```
state1 = digitalRead(button1);
```

Changes in void setup()

```
void setup()
{
    pinMode(4,OUTPUT);
    pinMode(5,OUTPUT);
    pinMode(6,OUTPUT);
    pinMode(7,OUTPUT);
    pinMode(8,OUTPUT);
    pinMode(9,OUTPUT);
    pinMode(10,OUTPUT);
    pinMode(11,OUTPUT);
    pinMode(button,INPUT);
    pinMode(button1,INPUT);
}
```

6. The state variable takes the state of increment button and state1 variable takes the state of decrement button.
7. If state is high then the number/count is increased and if state1 is high then number is decreased.
8. We also take care of underflow and overflow conditions in the loop function where if a number reaches -1 its reset to 99 and if a number reaches 100 its reset to 0.

Final Circuit Image :



Full Arduino Code is attached in the form of .ino file.

Filename: **group9_2digitcounter.ino**

Simulation:

Now we can simulate the counter logic

To simulate

- The count starts from 00 and ends at 99
- First button is for increasing the count
- Second button is for decreasing the count
- If count reaches 100 by increment its reset to 0
- If count reaches -1 by decrement its reset to 0.

Simulation video is attached with the parent folder.

For simulation,

Tinkercad project link - [Up down counter](#)