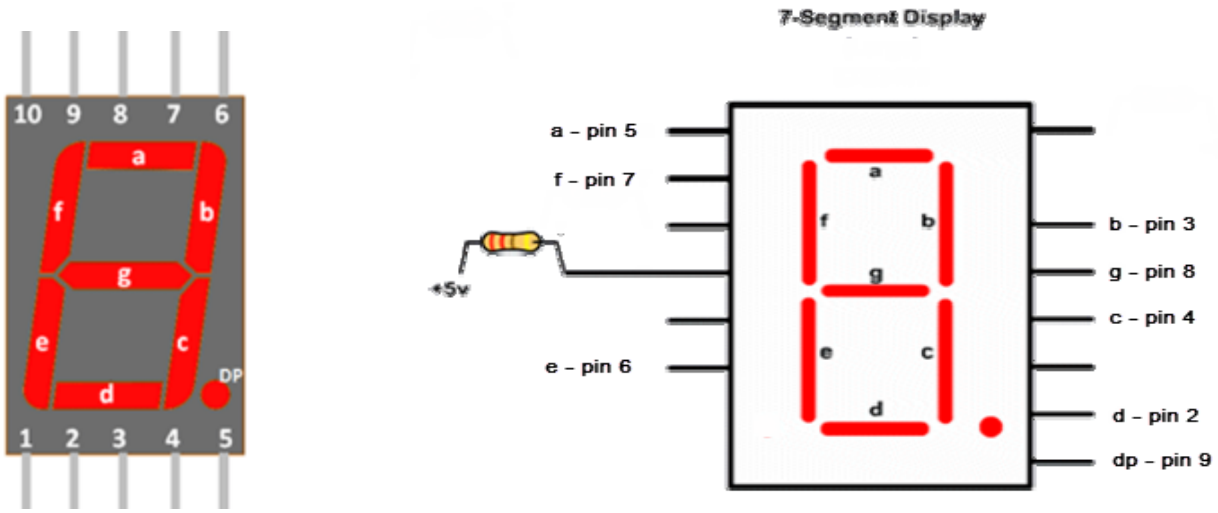
ARDUINO Lab 1: Countdown Timer with 7-Segment Display

A seven-segment display (SSD is a form of electronic display device for displaying decimal numerals that is an alternative to the more complex dot-matrix displays. Seven-segment displays are widely used in digital clocks, electronic meters, and other electronic devices for displaying numerical information. A seven segment display, as its name indicates, is composed of seven elements. Individually on or off, they can be combined to produce simplified representations of the Arabic numerals. The seven segments are arranged as a rectangle of two vertical segments on each side with one horizontal segment on the top, middle, and bottom. Additionally, the seventh segment bisects the rectangle horizontally. There are also fourteen-segment displays and sixteen-segment displays (to represent both numbers and letters)

HARDWARE REQUIRED

- 1 - Arduino Board
- 1 – Seven Segment Display (Common Anode)
- 3 - 220 Ω Resistor (or 330 Ω)
- 1 – DIP Switch

CIRCUIT:



Use the pinout diagram above to wire the seven-segment display to the appropriate pins on your Arduino board.

CODE

Enter the following code, check it and upload.

Sketch #1: This Arduino software example counts down from 9 to 0. This is a nice, compact version that uses a 2 dimensional array to hold the LED bit patterns, and "for" loops to get things done. Sketch 2 is the easier way to get the same effect but it involves more lines of code.

```
/* Always start off each new Arduino program with the following program header
Author:
Date:
Description:
*/

int segment_d = 2;
/* Segment_d is an example of a global variable. Any function within the code
can access it.Declare the other pin numbers used in the setup function below as
variables
here.  Example: instead of pinMode(2, OUTPUT), pinMode(segment_d, OUTPUT was
used instead.  Do this for pins 3 - 9
*/

byte seven_seg_digits[10][7] = //add a comment here that explains this
{
{ 0,0,0,0,0,0,1 },
{ 1,0,0,1,1,1,1 },
{ 0,0,1,0,0,1,0 },
{ 0,0,0,0,1,1,0 },
{ 1,0,0,1,1,0,0 },
{ 0,1,0,0,1,0,0 },
{ 0,1,0,0,0,0,0 },
{ 1,0,0,0,1,1,1 },
{ 0,0,0,0,0,0,0 },
{ 1,0,0,0,1,0,0 }
};
void setup()
{
      Serial.begin(9600);
      Serial.println("The program is starting");
      pinMode(segment_d, OUTPUT);
      pinMode(3, OUTPUT);
      pinMode(4, OUTPUT);
      pinMode(5, OUTPUT);
      pinMode(6, OUTPUT);
      pinMode(7, OUTPUT);
      pinMode(8, OUTPUT);
      pinMode(9, OUTPUT);
      writeDot(1);
}
void writeDot(byte dot){
```

```
      digitalWrite(9, dot);
}
void sevenSegWrite(byte digit) {
      byte pin = 2;
      for (byte segCount = 0; segCount < 7; ++segCount) {
            digitalWrite(pin, seven_seg_digits[digit][segCount]);
            ++pin;
      }
}

void loop()
{
      Serial.println("This is the top of the loop function");
      for (byte count = 10; count > 0; --count) {
            delay(1000);
            sevenSegWrite(count - 1);
      }
      delay(4000);
}
```

Program #1 (to be submitted):

Change Sketch #1 Above to Count Up

Change the code in sketch #1 above so that it counts up instead of down. Make sure that when the Arduino is powered on initially, that you do not see an "8" on the seven segment display. Instead, all segments should be off before the first number is shown.

Add comments throughout and Serial.println statements throughout your code

Once you verify that the code is working, comment the code fully so it is clear that you understand how it works. You may need to do some research on 2D arrays and new data types like *byte*. An example of the type of comment you should add is what the *writeDot* function does. Comment it out and run the program again to see what effect it has on the circuit. Then explain it in your code.

Use Serial.println() statements inside of the main loop function that prints out statements to the Serial monitor that clearly explains what is happening in the program everytime the number changes. This is helpful when troubleshooting code that does not work. Remember that digital I/O ports 0 and 1 are off limits when wiring your circuits if you are writing to the Serial monitor.

> Call your program *yourFirstName_7seg.ino* [You will be making changes to this file later on in this lab]

Sketch #2: Here is a version that does the same thing, but is easier to understand:

```
/* Always start off each new Arduino program with the following program
Author:
Date:
Description:
```

```
*/

// Declare specific variable names here that represent integers 2-9 that
// represent the pin numbers in the setup function.

void setup() {
      pinMode(2, OUTPUT);
      pinMode(3, OUTPUT);
      pinMode(4, OUTPUT);
      pinMode(5, OUTPUT);
      pinMode(6, OUTPUT);
      pinMode(7, OUTPUT);
      pinMode(8, OUTPUT);
      pinMode(9, OUTPUT);
      digitalWrite(9, 1); // start with the "dot" off
}
void loop() {
// write '9'
digitalWrite(2, 1);
digitalWrite(3, 0);
digitalWrite(4, 0);
digitalWrite(5, 0);
digitalWrite(6, 1);
digitalWrite(7, 0);
digitalWrite(8, 0);
delay(1000);
// write '8'
digitalWrite(2, 0);
digitalWrite(3, 0);
digitalWrite(4, 0);
digitalWrite(5, 0);
digitalWrite(6, 0);
digitalWrite(7, 0);
digitalWrite(8, 0);
delay(1000);
// write '7'
digitalWrite(2, 1);
digitalWrite(3, 0);
digitalWrite(4, 0);
digitalWrite(5, 0);
digitalWrite(6, 1);
digitalWrite(7, 1);
digitalWrite(8, 1);
delay(1000);
// write '6'
digitalWrite(2, 0);
digitalWrite(3, 1);
digitalWrite(4, 0);
digitalWrite(5, 0);
digitalWrite(6, 0);
digitalWrite(7, 0);
digitalWrite(8, 0);
```

```
delay(1000);
// write '5'
digitalWrite(2, 0);
digitalWrite(3, 1);
digitalWrite(4, 0);
digitalWrite(5, 0);
digitalWrite(6, 1);
digitalWrite(7, 0);
digitalWrite(8, 0);
delay(1000);
// write '4'
digitalWrite(2, 1);
digitalWrite(3, 0);
digitalWrite(4, 0);
digitalWrite(5, 1);
digitalWrite(6, 1);
digitalWrite(7, 0);
digitalWrite(8, 0);
delay(1000);
// write '3'
digitalWrite(2, 0);
digitalWrite(3, 0);
digitalWrite(4, 0);
digitalWrite(5, 0);
digitalWrite(6, 1);
digitalWrite(7, 1);
digitalWrite(8, 0);
delay(1000);
// write '2'
digitalWrite(2, 0);
digitalWrite(3, 0);
digitalWrite(4, 1);
digitalWrite(5, 0);
digitalWrite(6, 0);
digitalWrite(7, 1);
digitalWrite(8, 0);
delay(1000);
// write '1'
digitalWrite(2, 1);
digitalWrite(3, 0);
digitalWrite(4, 0);
digitalWrite(5, 1);
digitalWrite(6, 1);
digitalWrite(7, 1);
digitalWrite(8, 1);
delay(1000);
// write '0'
digitalWrite(2, 0);
digitalWrite(3, 0);
digitalWrite(4, 0);
digitalWrite(5, 0);
digitalWrite(6, 0);
```

```
digitalWrite(7, 0);
digitalWrite(8, 1);
delay(4000);
}
```

EXTENSION: You will be modifying your ***FirstName_7seg.ino*** file.

Wire two square push buttons on your breadboard. One button will hold the count steady. This means that if the current digit displayed is "2" and then you press the first button, the number "2" will stay on the display until you release the button (*hint: consider using a WHILE loop*). The second button will reset the count back to 0 when it is pressed. Once you release the button, the count will continue. ***You will need to use external interrupts to make this happen. Use pins 2 and 3 on the Arduino for this purpose. Pin 2 will connect to one of the push buttons and pin 3 will connect to the other push button. Do some research and see if you can figure it out.***

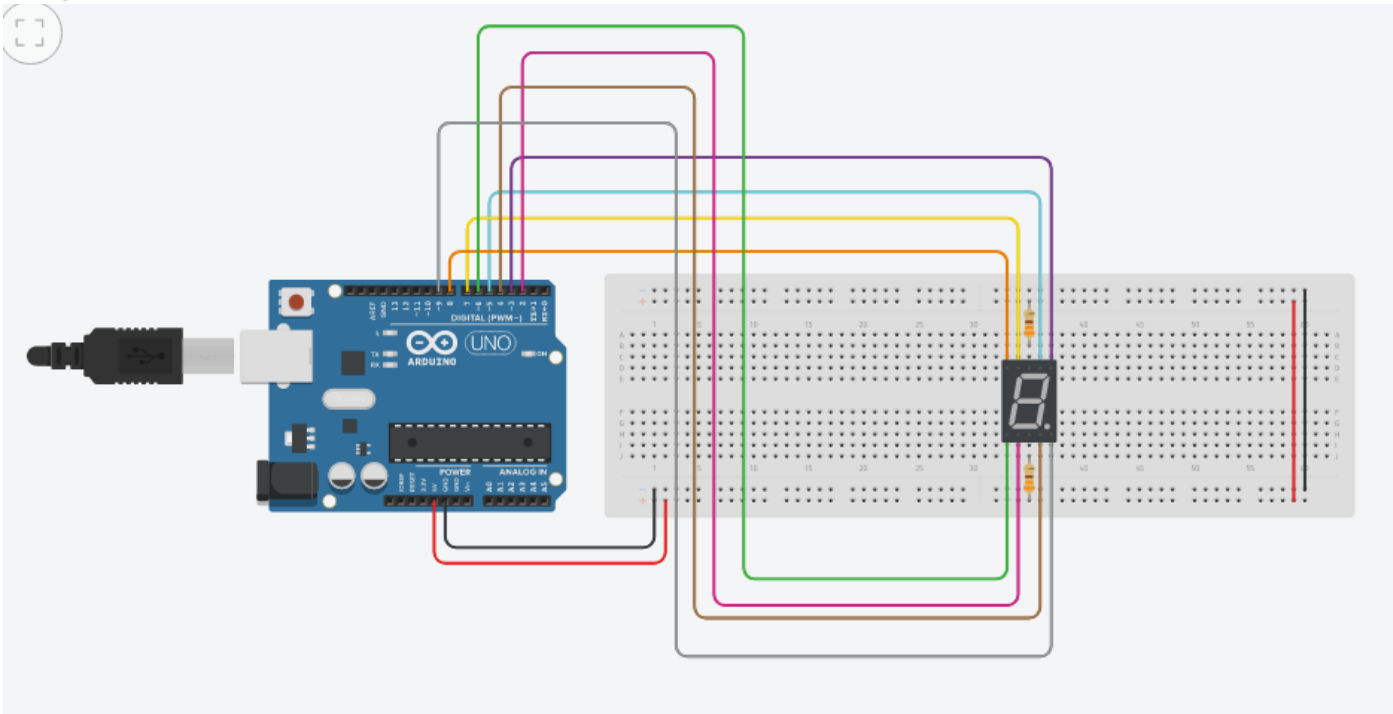
**** Remember to comment your code fully and use print statements throughout so you can follow the execution of your program ****

> Call your program *yourFirstName_7seg2.ino*

In this assignment post, upload your Arduino files (.INO) file. a video of the circuit working, and a shared link to your circuit.

ANSWERS

Program #1

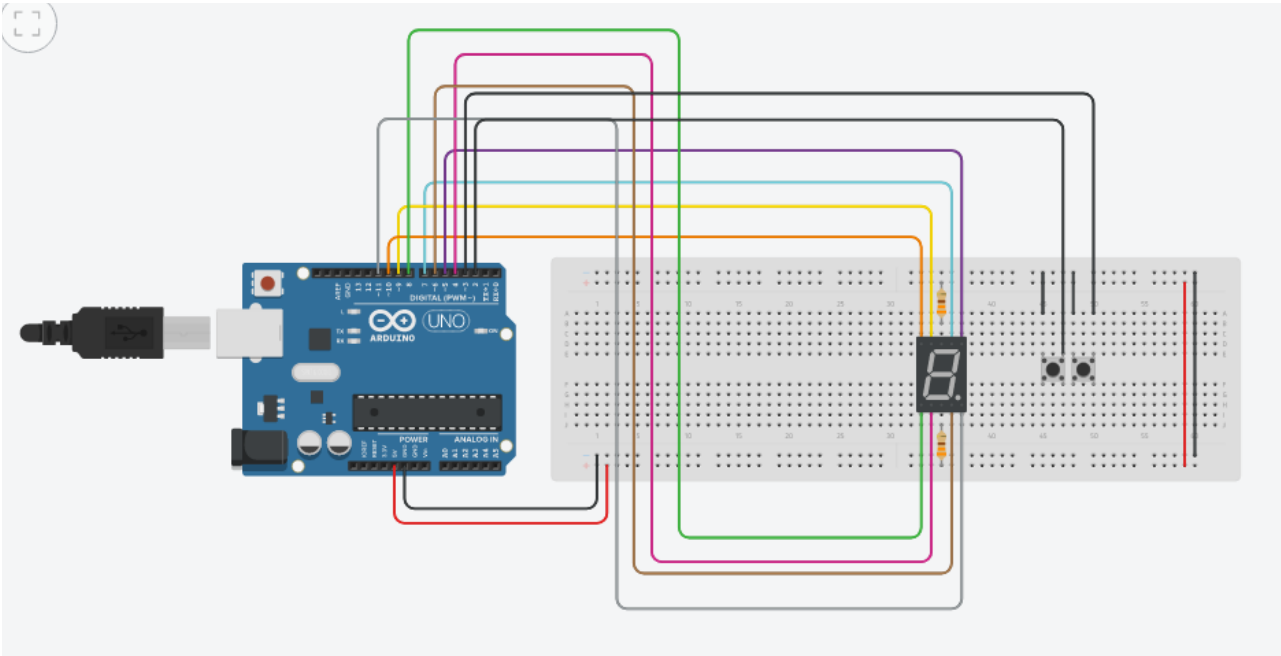



```
/*
Author: Gurpreet Singh
Date:2021-04-17
Description: This circuit counts up from 0 to 9 on the 7 segment display
*/

int segment_d = 2;
/* Segment_d is an example of a global variable. Any function within the code
can access it.Declare the other pin numbers used in the setup function below as
variables
here.  Example: instead of pinMode(2, OUTPUT), pinMode(segment_d, OUTPUT was
used instead.  Do this for pins 3 - 9
*/

byte seven_seg_digits[10][7] = //Array that stores the output of each pin for a
specific number to be shown on display
{
{ 0,0,0,0,0,0,1 }, //Displays 0
{ 1,0,0,1,1,1,1 }, //Displays 1
{ 0,0,1,0,0,1,0 }, //Displays 2
{ 0,0,0,0,1,1,0 }, //Displays 3
{ 1,0,0,1,1,0,0 }, //Displays 4
{ 0,1,0,0,1,0,0 }, //Displays 5
{ 0,1,0,0,0,0,0 }, //Displays 6
{ 1,0,0,0,1,1,1 }, //Displays 7
```

```
{ 0,0,0,0,0,0,0 }, //Displays 8
{ 1,0,0,0,1,0,0 }  //Displays 9
};
void setup()
{
Serial.begin(9600); //Speed of the program (bits/seconds)
Serial.println("The program is starting");
//Initializes all segment pins to "OUTPUT"
pinMode(segment_d, OUTPUT);
pinMode(3, OUTPUT);
pinMode(4, OUTPUT);
pinMode(5, OUTPUT);
pinMode(6, OUTPUT);
pinMode(7, OUTPUT);
pinMode(8, OUTPUT);
pinMode(9, OUTPUT);
//Keeps all the segments off in the beginning of the program (prevents 8 from
displaying)
digitalWrite(2,HIGH);
digitalWrite(3,HIGH);
digitalWrite(4,HIGH);
digitalWrite(5,HIGH);
digitalWrite(6,HIGH);
digitalWrite(7,HIGH);
digitalWrite(8,HIGH);
digitalWrite(9,HIGH);
writeDot(1);
}
//This function receives either 0/1 and outputs LOW (0) OR high(1)for DP
void writeDot(byte dot){
digitalWrite(9, dot);
}
//This function recieves a row number and reads each column for that specific
row in the seven_seg_digits array
void sevenSegWrite(byte digit) {
byte pin = 2;
//Reads each column of a specific row in order to send HIGH to the appropriate
pin
for (byte segCount = 0; segCount < 7; ++segCount) {
digitalWrite(pin, seven_seg_digits[digit][segCount]);
++pin;
}
}

void loop()
{
Serial.println("This is the top of the loop function");
//This for loop keeps track of the rows in the seven_seg_digits array
for (byte count = 0; count < 10; ++count) {
  delay(1000); //Delays for 1000 millisecond
  sevenSegWrite(count); //Sends count to the sevenSegWrite function
}
```

```
delay(4000); //Delays for 4000 milliseconds
}
```

Extension



```
/*
  Author: Gurpreet Singh
  Date: 2021-04-19
  Description: This program extends the previous lab and adds 2 pushbutton that
hold and reset the display.
*/

int segment_d = 4;
int segment_e = 8;
int segment_c = 6;
int segment_b = 5;
int segment_a = 7;
int segment_f = 9;
int segment_g = 10;
int segment_p = 11;
int dip_one = 2; //For the first Dip Switch
int dip_two = 3; //For the second Dip Switch
int count = 0; //made it a globle variable so that it can be used in new
methods
boolean state = false; //used in the while loop


/*Segment_d is an example of a global variable. Any function within the code
can access it.Declare the other pin numbers used in the setup function below as
variables
```

```
  here.  Example: instead of pinMode(2, OUTPUT), pinMode(segment_d, OUTPUT
//was used instead.  Do this for pins 3 - 9
*/

byte seven_seg_digits[10][7] =
{
  { 0, 0, 0, 0, 0, 0, 1 },
  { 1, 0, 0, 1, 1, 1, 1 },
  { 0, 0, 1, 0, 0, 1, 0 },
  { 0, 0, 0, 0, 1, 1, 0 },
  { 1, 0, 0, 1, 1, 0, 0 },
  { 0, 1, 0, 0, 1, 0, 0 },
  { 0, 1, 0, 0, 0, 0, 0 },
  { 1, 0, 0, 0, 1, 1, 1 },
  { 0, 0, 0, 0, 0, 0, 0 },
  { 1, 0, 0, 0, 1, 0, 0 }
};
/*seven_seg_digits is a two dimensional array that is used to store the
information of LEDs
  for example when it needs to display 0, g-segment of the LED is turned off.
*/

void setup()
{
  Serial.begin(9600);
  Serial.println("The program is starting");
  pinMode(segment_d, OUTPUT);
  pinMode(segment_a, OUTPUT);
  pinMode(segment_b, OUTPUT);
  pinMode(segment_c, OUTPUT);
  pinMode(segment_e, OUTPUT);
  pinMode(segment_f, OUTPUT);
  pinMode(segment_g, OUTPUT);
  pinMode(segment_p, OUTPUT);
  pinMode(dip_one, INPUT);
  pinMode(dip_two, INPUT);
  pinMode(3, INPUT_PULLUP); //reads the dip switch, if its on or off
  pinMode(2, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(2), pause, CHANGE); //Interrupt for the
first dip switch, executes the method pause
  attachInterrupt(digitalPinToInterrupt(3), reset, CHANGE); //Interrupt for the
second dip switch, executes the method reset
  writeDot(1);
}
void pause() {
  Serial.println("Pausing the counter ");
  state = !state;
}
//stops the loop from continuing

void reset() {
  Serial.println("Reseting the counter");
```

```
  if (count != 0) {
    count = 0;
  }
  state = !state;
}
//Resets the counter and starts it from 0

/*The method setup() declares that pins 2-9 are used as outputs since they are
all LEDs
  A value of 1 is sent to the method writeDot(), this is used to turn all the
segments off when the program starts
*/
void writeDot(byte dot) {
  for (byte pin = 4; pin < 12; ++pin) {
    digitalWrite(pin, dot);
  }
}
/* Method writeDot recieves a byte parameter of 1, then a for loop is used to
go through all the pins and turn them off
  This is done so that when the program starts, it doesn't show "8"
*/

void sevenSegWrite(byte digit) {
  byte pin = 4; //starts the pin at 2 and goes up
  for (byte segCount = 0; segCount < 7; ++segCount) {
    digitalWrite(pin, seven_seg_digits[digit][segCount]);
    ++pin;
  }
}
/*
  The method sevenSegWrite recieves a byte as a parameter.
  It uses the for loop to turn on the LEDs, it retrieves the LED information
from the 2-D array we created
*/

void loop()
{
  Serial.println("The value of d3 is ");
  Serial.println(digitalRead(3));
  while (state == true) {
    delay(1000);
    sevenSegWrite(count);
    ++count;
    if (count == 10) { //resets the count once it reaches 10
      count = 0;
      delay(4000);// 4 second delay after every cycle
      Serial.println("The cycle is restarting");
    }
  }
}
/*loop() runs while the "state" is true, if any of the interrupts are
triggered, the state becomes false
```

```
  There is a 1 second delay after each number is displayed on the segment
  There is 4 second delay after every cycle, when all the values from 0-9 are
displayed
*/
```