NATIONAL INSTITUTE OF TECHNOLOGY SIKKIM

Embedded Systems
(EC16101)
Project Report
Academic Year 2023-2024

Submitted by:

Sonu Kumar Bhagat (B210066EC)

Semester VI

Submitted to:

Dr. Neha K. Nawandar

Contents:

1. Problem statement:

Design an entry number display system. The system must display the entry numbers of your group members one by one. You can either use an LCD or an SSD as a display. If you use SSD, keep a delay of at least 500ms between individual characters of one entry number and 2 seconds between consecutive entry numbers.

2. Hardware and software components:

Hardware Components	Software Components
Arduino UNO *1	Arduino IDE (Arduino Web Editor)
Resistors (330 ohms) * 7	
SSD (Common Cathode type) *1	
Arduino Cable *1	
Breadboard *1	
Male-to-Male Jumper Wires	

3. Circuit diagram/ design schematic:

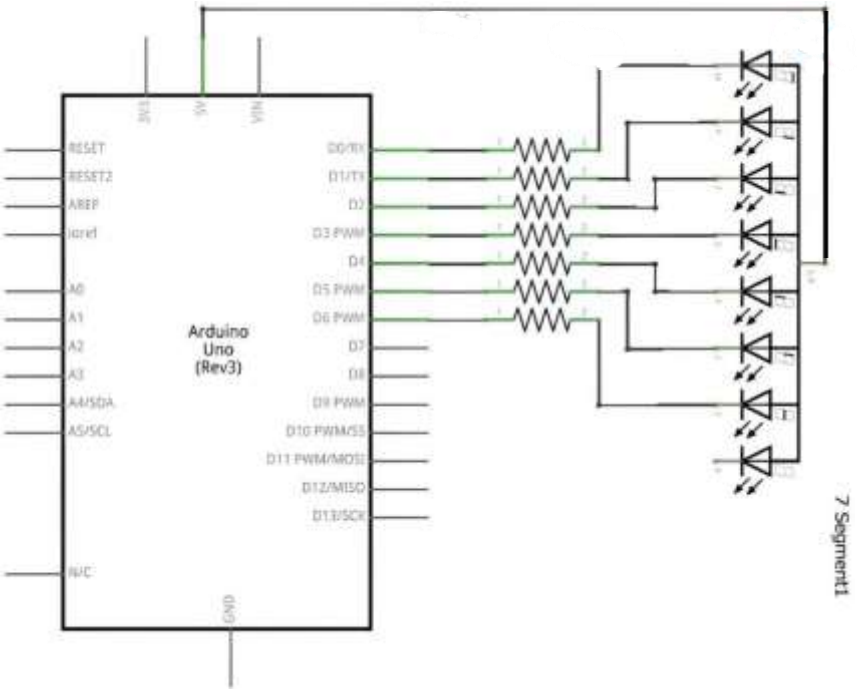
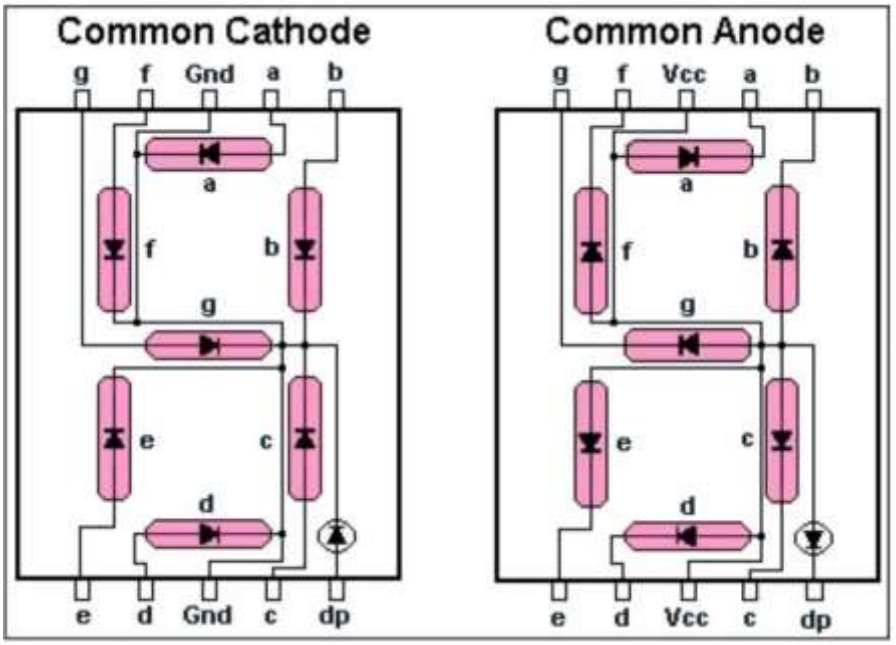


Fig.: Interfacing SSD with Arduino

Pin configuration of SSD (Seven Segment display):



4. Theory:

When interfacing with any kind of display device, Arduino and other controllers use digital output or serial communication. Driving SSDs by Arduino is as simple as driving an LED on it. The seven-segment display is made up of eight LED segments. These are used to display numbers.

Seven-segment displays (SSDs):

An SSD is a unit that is comprised of eight LED segments. Out of these eight LED segments, seven are bar-shaped and one is a dot.

There are 10 pins total on an SSD. Out of these 10 pins, two pins are common cathode or common anode, and the rest all are connected to opposite terminals of the LED segments.

SSDs are used to display numbers. They are less costly and often employed as an alternative to character LCDs. These are used in embedded applications where the display only needs to show numbers. Take a digital clock, for example, which displays the time in numbers. It can be built using SSDs rather than an expensive LCD.

A token display board can also be built using SSDs as it only needs to display token numbers. Similarly, SSDs can be used in digital dashboards (such as the one on vehicle dashboards), which display numeric information (distance traveled, fuel efficiency, etc.).

Additionally, SSDs are used in several embedded and consumer applications where numeric data is displayed.

A few examples include:

- The display timer for a microwave or a washing machine
- The temperature gauge on a space heater or air-conditioner
- The amount for a currency counting machine
- The numbers on a calculator, etc.

Types of SSDs:

There are two types of SSDs:

1. Common-anode
2. Common-cathode

In the common-anode SSD, the anode of each of the LED segments has a common terminal while cathodes have separate terminals. To control each segment, the common anode is connected to the VCC and the cathode terminals are displayed as “logical LOW” to turn ON the segments and “logical HIGH” to turn them OFF.

In the common-cathode SSD, the cathodes if each of the LED segments has a common terminal while the anodes have separate terminals. To control each segment, the common cathode is connected to ground and the anode terminals are displayed as “logical HIGH” to turn ON the segments and “logical LOW” to turn them OFF.

How SSD works?

There are eight LED segments on an SSD, of which seven are bar-shaped and one is dot-shaped. The bar-shaped LED segments are designated by the alphabet letters “A” to “G” and the dot-shaped LED segment is designated by “DP.”

By lighting specific combinations of the LED segments, different decimal digits (0 to 9) and hexadecimal alphabets (A to F) can be displayed.

In the common-anode SSD, the common terminal is the anode of each of the LED-segments. To display different digits and hexadecimal alphabets, the LED segments of the common-anode SSD must be provided as digital logic.

In the common-cathode SSD, the common terminal is the cathode with all of the LED-segments. To display different digits and hexadecimal alphabets, the LED segments of the common-cathode SSD must be provided as a digital logic.

To protect the LED segments, it is advisable to connect current-limiting resistors in series with each segment.

However, different LED segments may have different forward bias voltage. So, the resistors must be connected along each segment rather than connecting a single resistor to the common terminal. If a single resistor is connected to the common terminal, some of the LED segments may not glow because they may have a higher forward voltage. Alternatively, all of the segments may glow but with different light intensity.

The value of the resistor must be calculated according to the segment with the highest forward voltage.

The value of the resistor can be calculated using this equation:

RSeries = (VS – VLED)/ILED
Where,
RSeries = Value of resistor
VS = Source voltage
VLED = Highest forward voltage of LED segments
ILED = Current through LED segments

Let's suppose the highest forward voltage is 1.7V, the source voltage is 5V, and the current required through the LED segments is 5 mA to 20 mA — or take the average 10 mA.

Then, the value of resistors will be:

R = (5-1.7)/10×10-3
= 330 Ohms

Driving SSD with Arduino:

Interfacing SSD with Arduino is as simple as interfacing LEDs. The SSD can be treated like a collection of eight LEDs with a common anode or common cathode.

The instructions:

- For driving the common-anode SSD, the common terminal is connected to the VCC and the rest of the terminals are directly interfaced with Arduino's digital I/O channels via series resistors. The channels must be set to LOW to turn ON the LED segments, and they must be set to HIGH to turn them OFF.
- For driving the common-cathode SSD, the common terminal is connected to the ground and the rest of the terminals are directly interfaced with Arduino's digital I/O channels via the series resistors. In this case, the channels must be set to HIGH to turn ON the LED segments, and they must be set to LOW to turn them OFF.The digital I/O channels of Arduino output 5V/3.3V, and sources or sinks current up to 40 mA, which is sufficient to drive the LED segments. By turning the different combinations of LED segments ON and OFF, different digits (0 to 9) and alphabets (A to F) can be displayed.
- For driving more than one SSD together, a multiplexing technique is used. When there are many SSDs that have to be driven by a controller, the SSDs are interfaced via a demultiplexer or BCD-to-seven-segment decoder IC, such as SN7446AN.

5. Codes:

```
// Pin Definitions for SSD
#define SSD_A 2
#define SSD_B 3
#define SSD_C 4
#define SSD_D 5
#define SSD_E 6
#define SSD_F 7
#define SSD_G 8
#define SSD_DP 9

// Array of SSD digit patterns for each number 0-9
byte digitPatterns[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
};

// Define the pins for each SSD segment
int segmentPins[] = {SSD_A, SSD_B, SSD_C, SSD_D, SSD_E, SSD_F, SSD_G};

void setup() {
  // Set SSD pins as OUTPUT
  for (int i = 0; i < 7; i++) {
    pinMode(segmentPins[i], OUTPUT);
  }
}

void loop() {
  // Display entry numbers one by one
  char* entryNumbers[] = {"40", "49", "66", "76"}; // Entry numbers: 40, 49, 66, 76
  int numEntries = sizeof(entryNumbers) / sizeof(entryNumbers[0]);
```

```
 for (int i = 0; i < numEntries; i++) {
    displayEntryNumber(entryNumbers[i]);
    delay(2000); // 2 seconds delay between consecutive entry numbers
    clearDisplay();
    delay(50); // Delay for blanking the display
  }
}

void displayEntryNumber(char* entryNumber) {
  int len = strlen(entryNumber);
  for (int i = 0; i < len; i++) {
    displayNumber(entryNumber[i] - '0');
    delay(500); // 500ms delay between individual characters
    clearDisplay();
    delay(50); // Delay for blanking the display
  }
}

void displayNumber(int num) {
  // Set the segments according to the digit pattern
  for (int i = 0; i < 7; i++) {
    digitalWrite(segmentPins[i],  digitPatterns[num][i]);
  }
}

void clearDisplay() {
  // Turn off all segments
  for (int i = 0; i < 7; i++) {
    digitalWrite(segmentPins[i], LOW); // Common Cathode, so setting LOW turns OFF
  }
}
```

6. Results:

The implementation of the entry number display system utilizing a Seven Segment Display with common cathode and Arduino Uno was successful. The system effectively displayed designated numbers, validating the correct integration of hardware components and the accuracy of the programmed control logic. This outcome affirms the feasibility of the designed solution for entry number display applications, showcasing the reliable performance of the implemented hardware-software interface.

Supporting Link:

https://drive.google.com/drive/folders/1068MRlDp01Z0QlopMsMEx4t681R2Ivcm

7. Discussion and Conclusion:

To display different digits on the SSD, different combinations of the LED segments are switched ON and OFF at a time. The SSD used in this project is a common-anode type. So, the common terminal of the SSD is connected to 5V DC and the rest of the terminals are interfaced with the Arduino channels via series resistors of 330 Ohms.

To display different digits, the logical table for a common-cathode seven-segment is applicable.

For flashing digits 0 to 9, the "0" is first displayed by turning the ON LED segments A, B, C, D, E, and F — while the LED segment G is turned OFF. After displaying "0" for one second, all of the LED segments are turned OFF for 300 milliseconds. Note: if all of the segments are not turned OFF, the LED segments will remain turned ON once they are switched back ON, and the successive digits will not display properly.

After displaying "0," if all of the LED segments are not turned OFF, when displaying "1," the SSD will still display "0." It will display "8" when 2, 3, 4, 5, 6, 8, and 9 are flashed, and then "0" when the 7 is flashed. So, it is important to turn OFF all of the LED segments for a few milliseconds after displaying each digit.

So, for example:

- For displaying “1,” the LED segments B and C are turned ON while the rest all are turned OFF.
- For displaying “2,” the LED segments A, B, G, E, and D are turned ON while rest all are turned OFF.
- For displaying “3,” the LED segments A, B, G, C, and D are turned ON while rest all are turned OFF.
- For displaying “4,” the LED segments F, G, B, and C are turned ON while rest all are turned OFF.
- For displaying “5,” the LED segments A, F, G, C, and D are turned ON while rest all are turned OFF.
- For displaying “6,” the LED segments A, F, G, E, D, and C are turned ON while rest all are turned OFF.
- For displaying “7,” the LED segments A, B, and C are turned ON while rest all are turned OFF.
- For displaying “8,” all of the LED segments (except DP) are turned ON.
- For displaying “9,” the LED segments A, B, F, G, and C are turned ON while rest all are turned OFF.

All of the LED segments are turned OFF for 300 milliseconds after displaying a digit for one second. As the Arduino keeps repeating its code, the display of the digits 0 to 9 will keep repeating until Arduino is shut down.