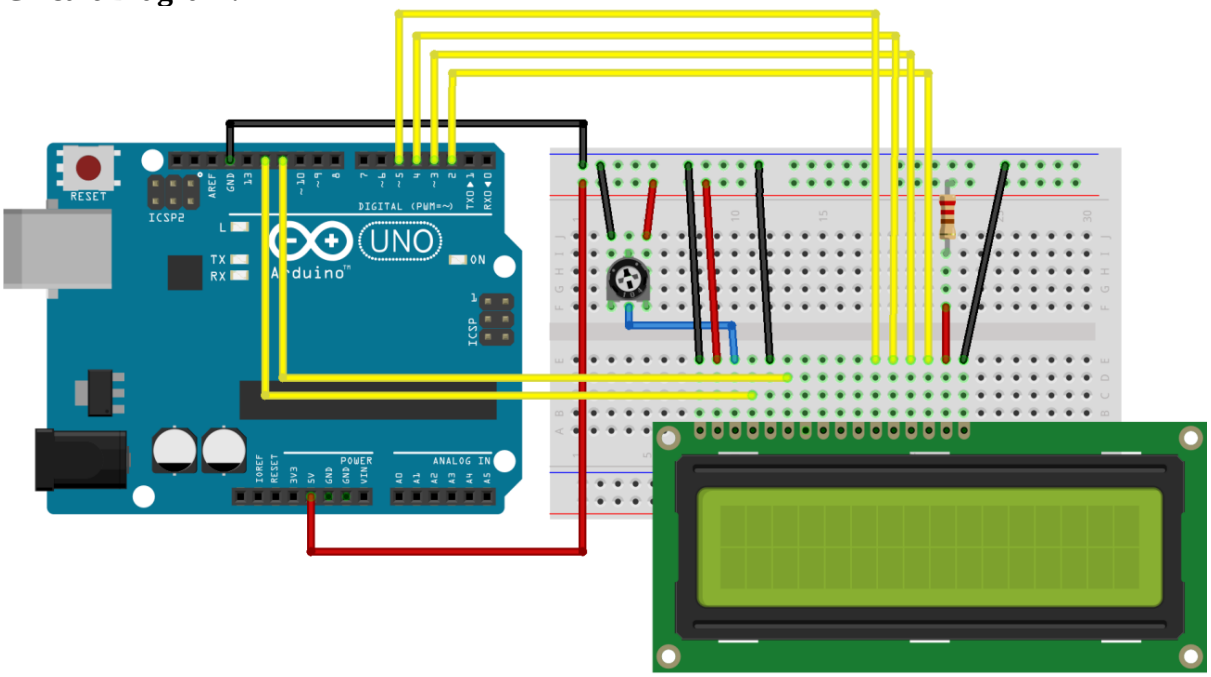
LCD interfacing

Aim: Our goal in this project is to scroll input text on LCD from right to left.
Software: Arduino IDE

Components Required: 1. System -1
2. Arduino Uno Board -1
3. Arduino dumping cable -1
4.16x2 LCD Screen
5. Potentiometer 10kΩ -1
6. Resistor 220Ω -1
7. Breadboard-1
8. Connecting Wires -Required

Theory: LCD (liquid crystal display) is the technology used for displays in notebook and other smaller computers. Like light-emitting diode (LED) and gas-plasma technologies, LCDs allow displays to be much thinner than cathode ray tube (CRT) technology.

Circuit Diagram:



Procedure:
1. Open Arduino IDE.
2. Write the code in the text editor.
3. Save the sketch with .ino extension.
4. Connect the hardware circuit and Connect your Arduino Board to the USB port of your computer.
5. Select the serial device of the Arduino board from the Tools | Serial Port menu.
6. Compile the file by clicking on the verify button.
7. If successful, the message "Done Compiling." will appear in the status bar.
8. If there is any errors it will show them in Transcript window, rectify those errors and compile it again.
9. Push the reset button on the board then click the Upload button in the IDE. Wait a

few seconds. If successful, the message "Done uploading." will appear in the status bar.

Code:

```
// include the library code:
#include <LiquidCrystal.h>
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup() {
// set up the LCD's number of columns and rows:
lcd.begin(16, 2);
// Print a message to the LCD.
lcd.print("hello, world!");
delay(1000);
}
void loop() {
// scroll 13 positions (string length) to the left
// to move it offscreen left:
for (int positionCounter = 0; positionCounter < 13; positionCounter++) {
// scroll one position left:
lcd.scrollDisplayLeft();
// wait a bit:
delay(150);
}
60
}
```

Result:-The input text is scrolled from right to left in LCD.

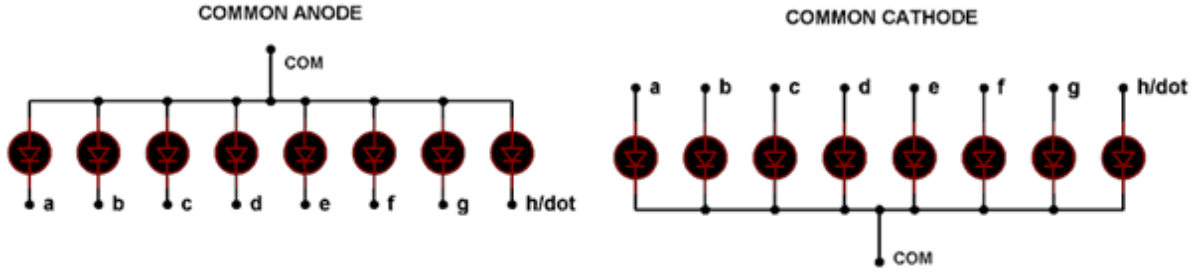
Segment Display Interfacing with Arduino

Aim:-In this topic we are going to interface a seven segment display to ARDUINO UNO. The display counts from 0-9 and resets itself to zero. Before going further, let us first discuss about seven segment displays.
Software: Arduino IDE

Components Required:

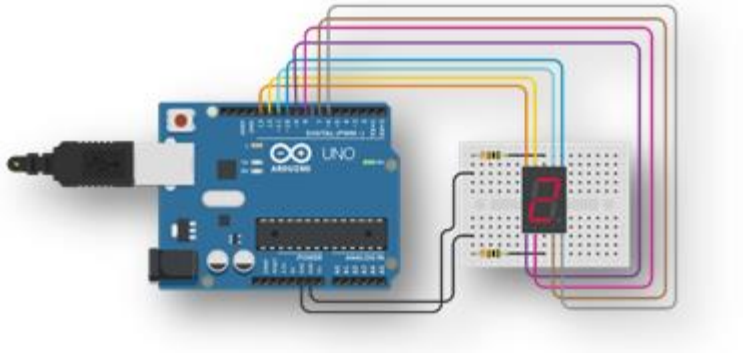
1. System -1
2. Arduino Uno Board -1
3. Arduino dumping cable -1
4. 7 Segment display
5. Potentiometer 10kΩ -1
6. Resistor 220Ω -1
7. Breadboard-1
8. Connecting Wires -Required
9. SN7446AN

Theory:-A seven-segment display got its name from the very fact that it got seven illuminating segments. Each of these segments has a LED. The LEDs are so fabricated that lighting of each LED is contained to its own segment. The important thing to notice here that the LEDs in any seven-segment display are arranged in common anode mode common positive) or common cathode mode (common negative).



Internal connections of 7 Segment Display

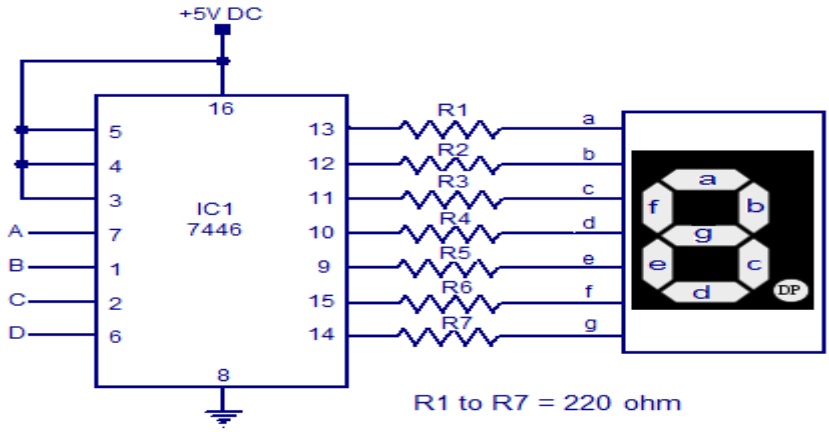
The circuit connection of LEDs in common cathode and common anode is shown in the above figure. Here one can observe that, in CC the negative terminals of every LED is connected together and brought out as GND. In CA the positive of every LED is connected together and brought out as VCC. These CC and CA come in very handy while multiplexing several cells together.



Using this method, you can drive the 7-segment display directly using the Arduino Uno. There are advantages and disadvantages to connecting a seven-segment display directly with the Arduino Uno. An advantage is that it eliminates the need for a special driver and thus reduces the cost. On the other hand, a direct connection requires more pins. And pin estate is a real concern for all engineers. So, we go for a drive.

SN7446AN (7seg display) driver:

Seven segment decoder / driver is a digital circuit that can decode a digital input to the seven-segment format and simultaneously drive a 7 segment LED display using the decoded information. What that will be displayed on the 7-segment display is the numerical equivalent of the input data. For example, a BCD to seven segment decoder drivers can decode a 4-line BCD (binary coded decimal) to 8-line seven segment format and can drive the display using this information. For example, if the input BCD code is 0001, the display output will be 1, for 0010 the display output will be 2 and so on. The circuit diagram shown below is of a BCD to seven segment decoder / driver using 7446 IC.

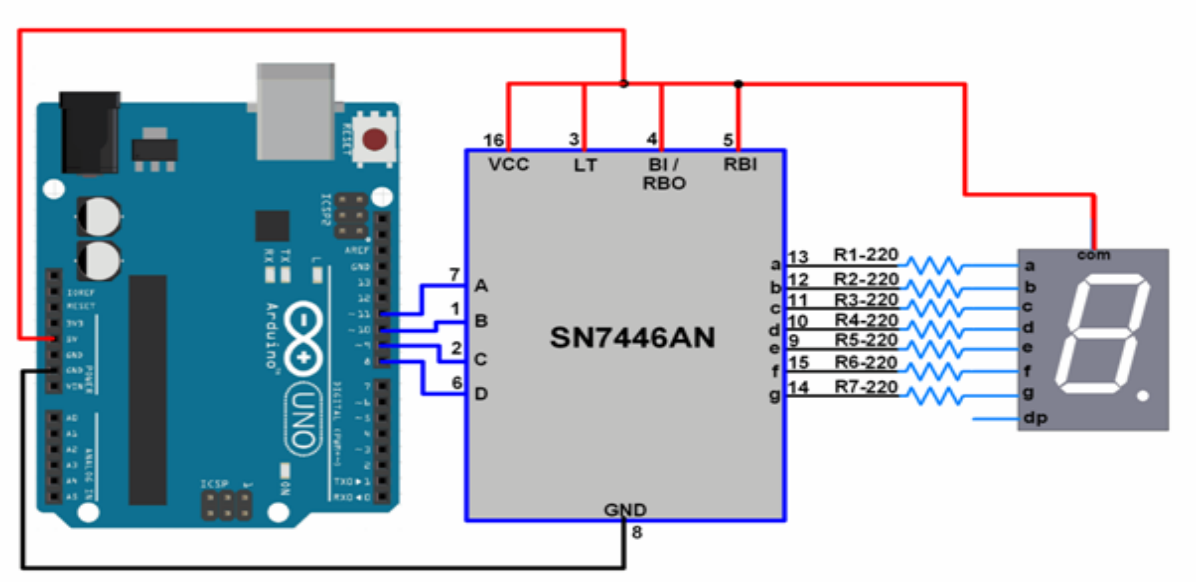


7446 seven segment decoder driver

7446 is a BCD to 7 segment display driver IC with active low outputs. The IC is stand alone and requires no external components other than the LED current limiting resistors. All outputs of the IC have complete ripple blanking and require no external driver transistors. There is also a built in lamp test function which can be used to test the LED segments. Pin 5 of the IC is the ripple blanking input (RBI) and pin 4 is the ripple blanking output (RBO). Pin 3 is the lamp test (LT) input pin. When the RBI and RBO pins are held high and the lamp test (LT) input pin 3 is held low all LED segment output goes high. The display used here must be a common anode type because the IC has active low outputs.

Interfacing 7-Segment Display Using SN7446AN Driver with Arduino UNO

Interfacing 7-Segment Display Using SN7446AN Driver with Arduino UNO



Code:

```
intbcd_pins[4]; /* array for A-D pins of driver IC */
 voidbcd_control_pins(int a, int b, int c, int d)        /* Assigns A-D pins of driver IC to Arduino board */
{
bcd_pins[0] = a;
bcd_pins[1] = b;
bcd_pins[2] = c;
bcd_pins[3] = d;
```

```
}
 voiddisplay_number(intnum)/* Function for displaying number (0-9) */
{
switch(num)
  {
case0:
digitalWrite(bcd_pins[0], LOW);              /* Drive bcd_pin[0] to LOW */
digitalWrite(bcd_pins[1], LOW);              /* Driving LOW turns on LED segment for
common anode display */
digitalWrite(bcd_pins[2], LOW);
digitalWrite(bcd_pins[3], LOW);
break;
case1:
digitalWrite(bcd_pins[0], HIGH);
digitalWrite(bcd_pins[1], LOW);
digitalWrite(bcd_pins[2], LOW);
digitalWrite(bcd_pins[3], LOW);
break;
case2:
digitalWrite(bcd_pins[0], LOW);
digitalWrite(bcd_pins[1], HIGH);
digitalWrite(bcd_pins[2], LOW);
digitalWrite(bcd_pins[3], LOW);
break;
case3:
digitalWrite(bcd_pins[0], HIGH);             /* Drive bcd_pin[3] to HIGH */
digitalWrite(bcd_pins[1], HIGH);             /* Driving HIGH turns on LED segment for
common anode display */
digitalWrite(bcd_pins[2], LOW);
digitalWrite(bcd_pins[3], LOW);
break;
case4:
digitalWrite(bcd_pins[0], LOW);
digitalWrite(bcd_pins[1], LOW);
digitalWrite(bcd_pins[2], HIGH);
digitalWrite(bcd_pins[3], LOW);
break;
case5:
digitalWrite(bcd_pins[0], HIGH);
digitalWrite(bcd_pins[1], LOW);
digitalWrite(bcd_pins[2], HIGH);
digitalWrite(bcd_pins[3], LOW);
break;
case6:
digitalWrite(bcd_pins[0], LOW);
digitalWrite(bcd_pins[1], HIGH);
digitalWrite(bcd_pins[2], HIGH);
digitalWrite(bcd_pins[3], LOW);
break;
case7:
digitalWrite(bcd_pins[0], HIGH);
```

```
digitalWrite(bcd_pins[1], HIGH);
digitalWrite(bcd_pins[2], HIGH);
digitalWrite(bcd_pins[3], LOW);
break;
case8:
digitalWrite(bcd_pins[0], LOW);
digitalWrite(bcd_pins[1], LOW);
digitalWrite(bcd_pins[2], LOW);
digitalWrite(bcd_pins[3], HIGH);
break;
case9:
digitalWrite(bcd_pins[0], HIGH);
digitalWrite(bcd_pins[1], LOW);
digitalWrite(bcd_pins[2], LOW);
digitalWrite(bcd_pins[3], HIGH);
break;
default:
digitalWrite(bcd_pins[0], LOW);
digitalWrite(bcd_pins[1], LOW);
digitalWrite(bcd_pins[2], LOW);
digitalWrite(bcd_pins[3], LOW);
break;
  }
}
 voidsetup() {
pinMode(8, OUTPUT);
pinMode(9, OUTPUT);
pinMode(10, OUTPUT);
pinMode(11, OUTPUT);
bcd_control_pins(11,10,9,8);            /* A-D of driver IC to Arduino */
}
 voidloop() {
inti;
for(i = 9; i>=0; i--)
  {
display_number(i);
delay(1000);
  }
for(i = 0; i<=9; i++)
  {
display_number(i);
delay(1000);
  }
}
```

Result:-Here, the 7-segment display is driven by the SN7446AN IC. It is a BCD to 7-segment driver/decoder IC. This reduces the number of pins required to drive the 7-segment display.