Lab no 02: Analog Control System Prototype LCD and Analog-to-Digital Interfaces

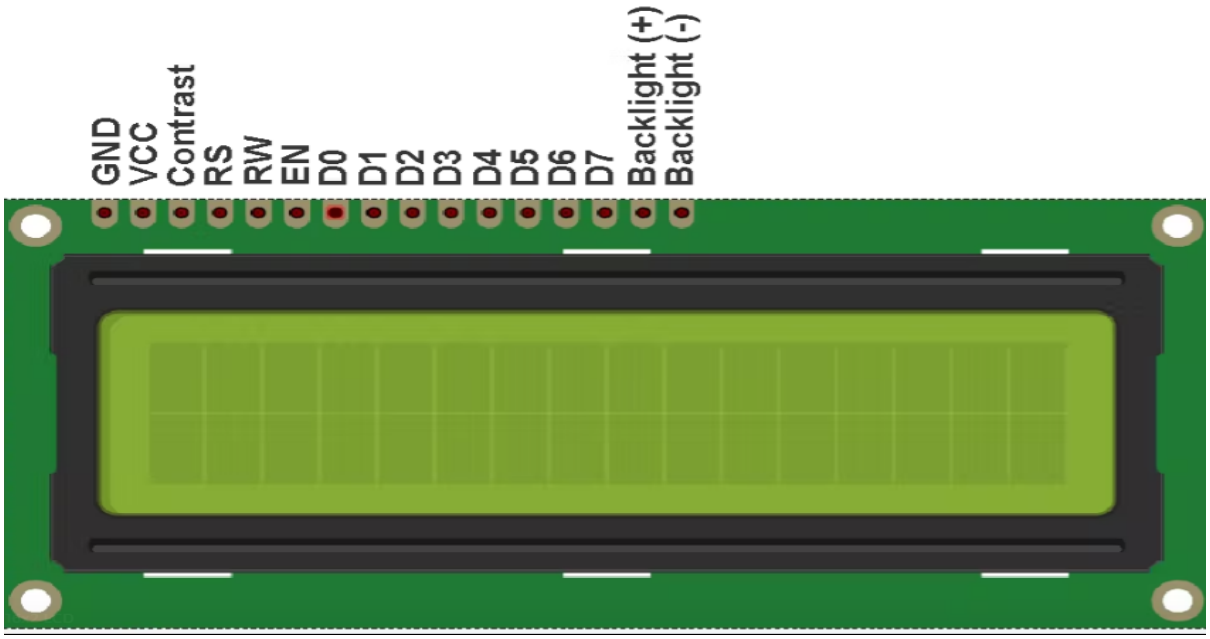
The purpose of this Lab is to learn interfaces with Liquid Crystal Display (LCD) and the Analog-to-Digital Converter (ADC). To do that, you are required to build a prototype for an analog control system prototype using a potentiometer.

You will use a potentiometer as input to Arduino, and the output will be displayed on LCD as solid square characters. As the voltage increases. The number of solid squares increases.

Parts: -

1. Introduction to LCD (16x2)
    a) Arduino and 16×2 LCD.
    b) Custom characters on 16×2 LCD.
    c) Hardware connections.
2. ADC Interface and potentiometer.
3. Analog control system prototype.

Part 1. Introduction to LCD (16x2)



Why is it called 16x2?

Because you can write 16 characters or numbers in column-wise and two in row-wise, this display has a total of 16 pins.

Pins functions are listed below: -

- GND(VSS)

  Connect the ground pin of the power supply to this pin.
- VCC

  Connect this pin to 5v
- Contrast (VEE)

  This pin is used to adjust the contrast of the Display. Connect a potentiometer (POT) to this pin. Rotate the knob of the POT to adjust the contrast.

- RS

  RS pin means Register select pin. Selects command register when the pin is LOW. And selects data register when this pin is HIGH.

- RW

  It represents the Read Write pin. When this pin is LOW, the MCU writes to register. And when the pin is HIGH, MCU reads from the register. Here we want to write. It connects it permanently to GND.

- EN (E)

  EN pin means the Enable pin. Send data to data pins when a HIGH to LOW pulse is given.

- D0-D7 (DB0-DB7)

  These are 8 data pins.

- Backlight (+)

  This is the anode pin of the backlight of the display

- Backlight (-)

  This is the cathode pin of the backlight of the display

Part 1. a) Arduino and 16×2 LCD.

Step - 1

Open Arduino IDE. Here we use the "Liquid Crystal" library. This is an inbuilt library, so there is no need to install that library separately. First, I am going to call this library.

```
#include <LiquidCrystal.h>
```

Step -2

Next, initialize the library with the number of interface pins with the function "LiquidCrystal lcd ()"

```
LiquidCrystal (rs, rw, enable, d0, d1, d2, d3, d4, d5, d6, d7)
```

parameters

- rs: the number of the Arduino pin that is connected to the RS pin on the LCD
- rw: the number of the Arduino pin that is connected to the RW pin on the LCD (*optional*)
- enable: the number of the Arduino pin that is connected to the enable pin on the LCD
- d0, d1, d2, d3, d4, d5, d6, d7: the numbers of the Arduino pins that are connected to the corresponding data pins on the LCD. d0, d1, d2, and d3 are optional; if omitted, the LCD will be controlled using only the four data lines (d4, d5, d6, d7).

Step - 3

Now can call this display by "lcd". Next, the program is the setup part. We need to set the number of columns and the number of rows. Here I am using the LCD with 16 columns and 2 rows. Set the number of columns and rows by the function, And set the A0 pin as an input.


```
lcd.begin(16, 2).
pinMode(A0,INPUT);
```

Step - 4

The setup part is over. Next loop part

First, I am going to clear the display. For that, I use the function "lcd. clear ()". This function will clean the entire display.

```
lcd.clear();
```

Step - 5

We need a starting point to start printing. So set the cursor to that particular point by the function "lcd.setCursor()".

This function has only two attributes. It is that starting point. (The number of columns and number of rows).

Here I am staring from the first column first row.

The first column is represented as 0, the second is 1, and so on, and the first raw is represented as 0, second is 1.

So, we need to start from the position (0, 0).

```
lcd.setCursor(0,0);
```

Step - 6

Next is the core part of this article—the instruction to print. I am going to print " Volume " by the instruction "lcd.print()". Alternatively, you can print anything. Please don, t forget the double quote marks.

```
lcd.print("Volume");
```

Step -7

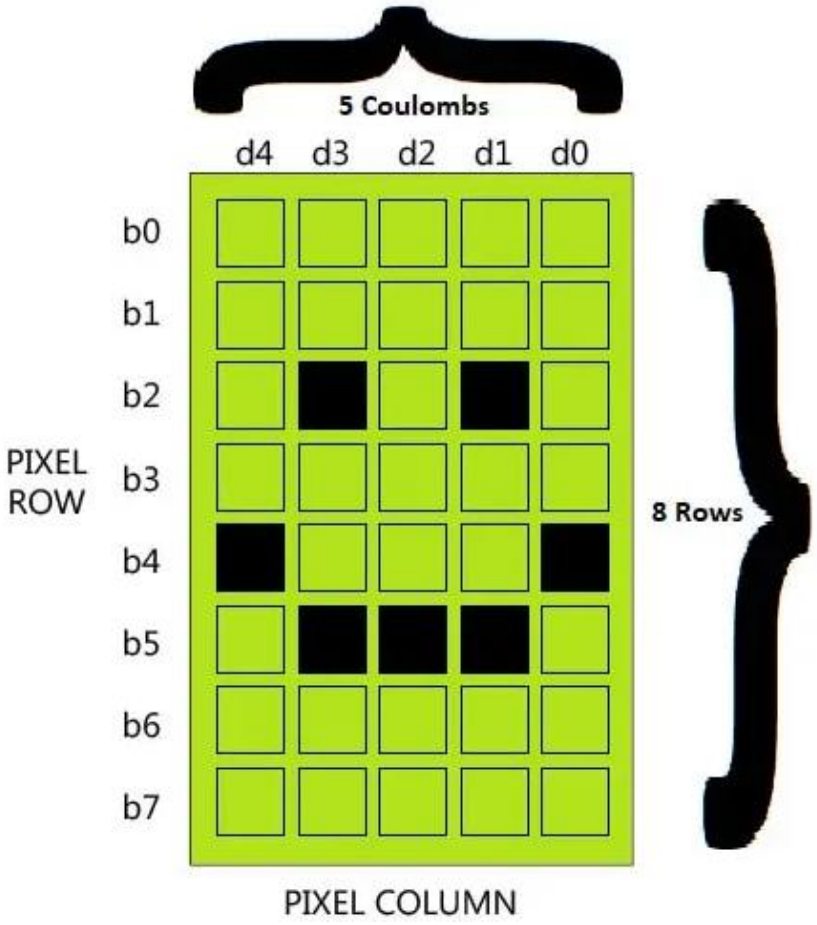
In the above printing statement, we use a total of 14 characters. So, the current position of the cursor is at (14, 0). Next, I am going to print on the next line, ie the position (0, 1). Set the cursor to that point by the function "lcd.setCursor()".

```
lcd.setCursor(0,1);
```

Part 1. b) Custom characters on 16×2 LCD.

16×2 LCD has an internal **CG-RAM**(character generated ram) in which we can generate or place our custom character. **CG-RAM** size is 64 bytes. We can generate/place **8 characters** of size **5×8** at a time in **CG-RAM**.

5×8 represents the dimension of the matrix in which a particular character can be displayed. 5 represents the number of coulombs and 8 represents the number of rows. 5×8 combined is a matrix size. Matrix is composed of pixels that we turn on and off to represent or make a character.



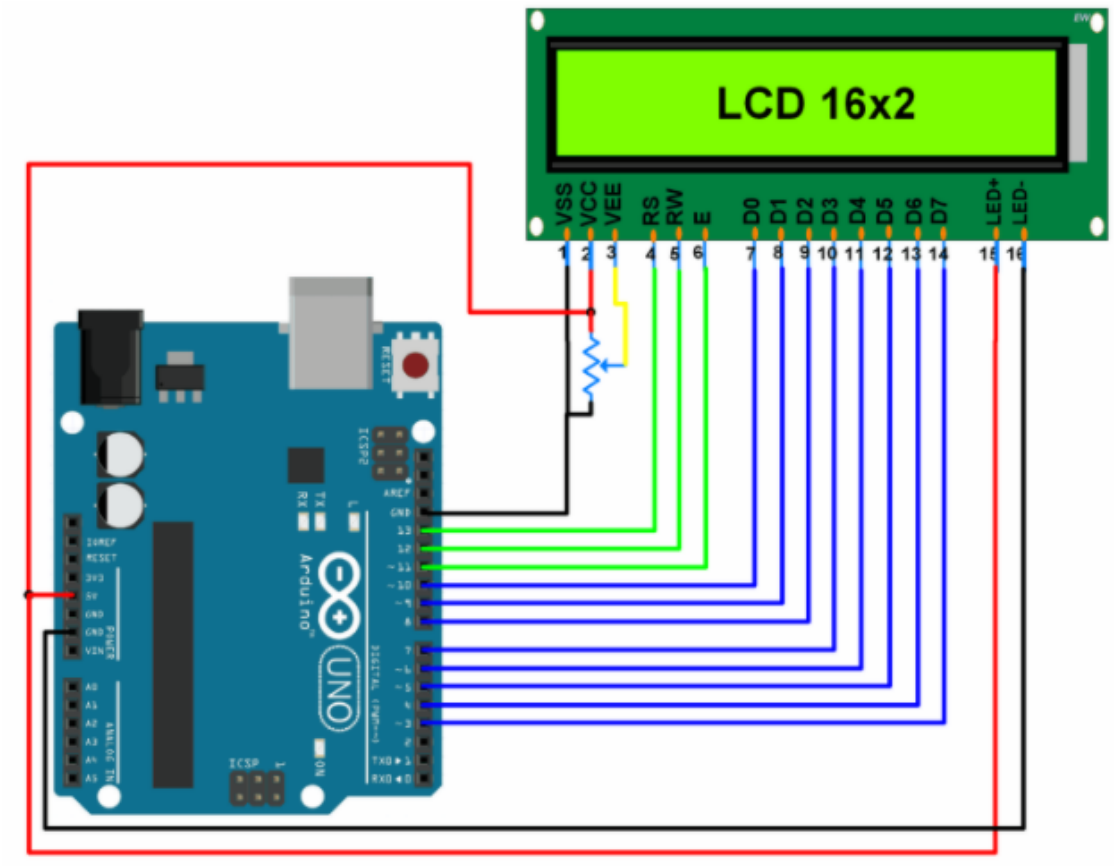
- Each switched on pixel binary value is ‘1‘.
- Each switched off pixel binary value is ‘0‘.

FOR Example :-

```
byte smiley[8] = {  B00000,B10001,B00000, B00000,
B10001, B01110, B00000,};
void setup() {
  lcd.createChar(0, smiley);
  lcd.begin(16, 2);
  lcd.write(byte(0));
}
void loop() {}
```

Part 1. c) Hardware Connection.

Connect LCD to Arduino as shown in the Figure below.



Part 2. ADC Interface and potentiometer.

When we interface sensors to the microcontroller, the output of the sensor many of the times is analog in nature. But microcontroller processes digital signals.

Hence, we use ADC in between sensor and microcontroller. It converts an analog signal into digital and gives it to the microcontroller.

There are many applications of ADC like biometric applications, Environment monitoring, Gas leakage detection etc.

Arduino Uno has 6 0n-board ADC channels which can be used to read analog signal in the range 0-5V.

It has a 10-bit ADC means it will give digital value in the range of 0 – 1023 (2^10). This is called as a resolution which indicates the number of discrete values it can produce over the range of analog values.

Digital Output value Calculation

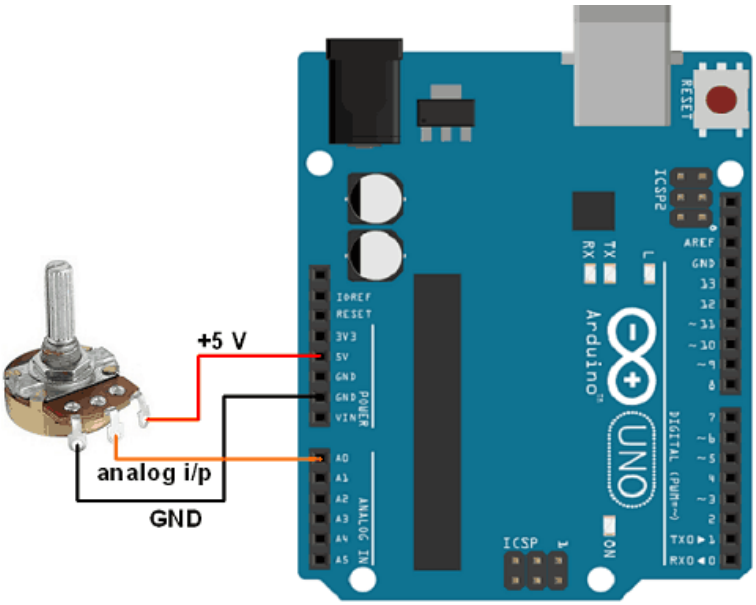
ADC Resolution = Vref / ((2^n) - 1)
Digital Output = Vin / Resolution

Where, **Vref -** The reference voltage is the maximum value that the ADC can convert.
To keep things simple, let us consider that Vref is 5V,

For 0 Vin, digital o/p value = 0

For 5 Vin, digital o/p value = 1023 (10-bit)

For 2.5 Vin, digital o/p value = 512 (10-bit)



Potentiometer connected Arduino ADC Channel

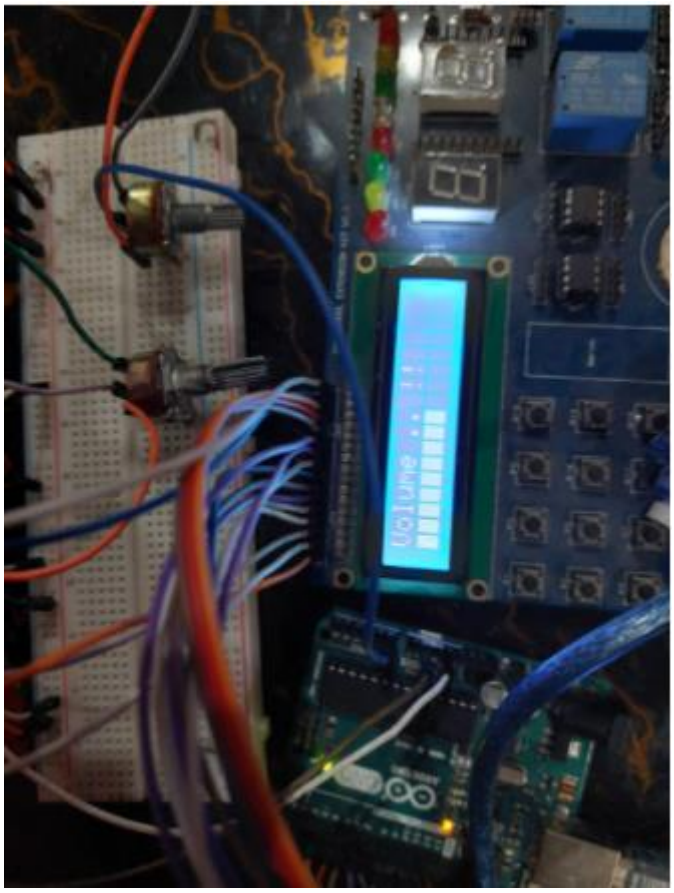
Use Potentiometer as analog Input

Connect Potentiometer (variable resistance) to Arduino as shown in the Figure above.

```
int sensorPin = A0; // input pin for the potentiometer
int digitalValue = 0;// variable to store the value
//coming from the sensor
void setup() {
  Serial.begin(9600);
}
void loop() {
// read the //value from the analog channel
  digitalValue = analogRead(sensorPin);
  Serial.print("digital value = ");
//print digital // value on serial monitor
  Serial.println(digitalValue);
  delay(1000);
}
```

Part 3. Analog control system prototype.

Build a prototype for an analog control system prototype using a potentiometer.
You will use a potentiometer as input to Arduino and the output will be displayed on LCD as solid square characters. As the voltage increases. the number of solid squares increases.



The analog control system prototype

```
#include <LiquidCrystal.h>
//LiquidCrystal(rs, rw, enable, d0, d1, d2, d3, d4, d5, d6, d7)
LiquidCrystal lcd(13,12,11,10,9,8,7,6,5,4,3);
```


```
int count=0;
//Custom characters byte arrays
byte
full[8]={B11111,B11111,B11111,B11111,B11111,B11111,B111
11,};
byte
empty[8]={B00000,B00000,B00000,B00000,B00000,B00000,B00
000,};
//Custom characters byte arrays
void setup()
{
lcd.begin(16 ,2);  //Initialize 16x2 lcd
pinMode(A0,INPUT);
lcd.clear();       //Clear lcd display screen
lcd.createChar(1 , full);   //Creating custom
characters in CG-RAM
lcd.createChar(2 , empty);
}

void loop()
{
 lcd.setCursor(0 ,0);            //Place lcd cursor on
first row and first coulomb of 16x2 lcd
  lcd.print("Volume .. !!");  //Display this test on
first row on 16x2 lcd
  int x= analogRead(A0);
  if(x<7)
  {
```


```
 lcd.setCursor(0 ,1);
 lcd.write(1);
 count=1;
}
else if(x>7 && x<13)
{
 lcd.setCursor(1 ,1);
 lcd.write(1);
 count=2;
}
else if(x>=13 && x<19)
{
 lcd.setCursor(2 ,1);
 lcd.write(1);
 count=3;
}
else if(x>=19 && x<25)
{
 lcd.setCursor(3 ,1);
 lcd.write(1);
 count=4;
}
else if(x>=25 && x<31)
{
 lcd.setCursor(4 ,1);
 lcd.write(1);
 count=5;
 }
```


```
else if(x>=31 && x<37)
{
 lcd.setCursor(5 ,1);
 lcd.write(1);
 count=6;
 }
 else if(x>=37 && x<42)
{
 lcd.setCursor(6 ,1);
 lcd.write(1);
 count=7;
 }
 else if(x>=42 && x<48)
{
 lcd.setCursor(7 ,1);
 lcd.write(1);
 count=8;
 }
 else if(x>=48 && x<54)
{
 lcd.setCursor(8 ,1);
 lcd.write(1);
 count=9;
 }
else if(x>=54 && x<60)
{
 lcd.setCursor(9 ,1);
 lcd.write(1);
```


```
 count=10;
 }
 else if(x>=60 && x<66)
{
 lcd.setCursor(10 ,1);
 lcd.write(1);
 count=11;
 }
 else if(x>=66 && x<72)
{
 lcd.setCursor(11 ,1);
 lcd.write(1);
 count=12;
 }
 else if(x>=72 && x<78)
{
 lcd.setCursor(12 ,1);
 lcd.write(1);
 count=13;
 }
 else if(x>=78 && x<84)
{
 lcd.setCursor(13 ,1);
 lcd.write(1);
 count=14;
 }
 else if(x>=84 && x<89)
{
```


```
   lcd.setCursor(14 ,1);
   lcd.write(1);
   count=15;
   }
  else if(x>=89 && x<95)
  {
   lcd.setCursor(15 ,1);
   lcd.write(1);
   count=16;
   }
   else if(x>=95 && x<100)
  {
   lcd.setCursor(16 ,1);
   lcd.write(1);
   count=16;
   }

      for(int i=count;i<=16;i++)
      {
          lcd.setCursor(i ,1);
          lcd.write(2);
        }
}
```