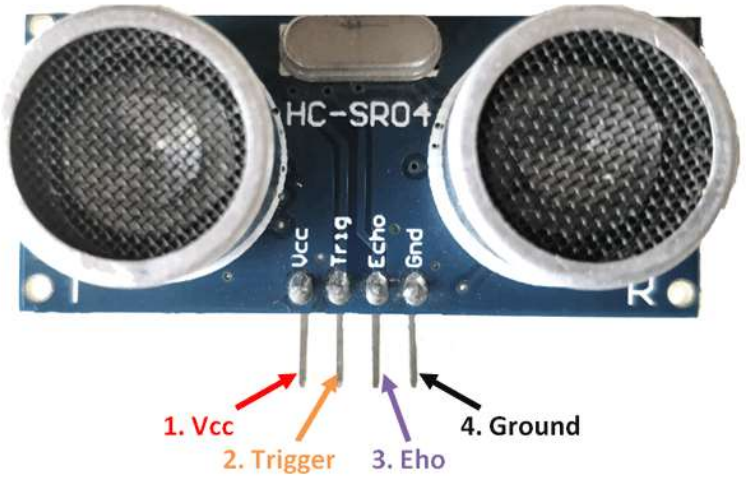
Experiment No. 8

Aim: To understand the working and programming of Ultrasonic Sensor.

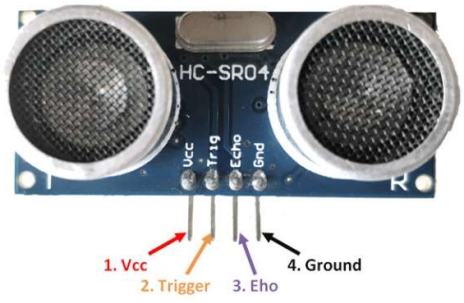
Theory

Ultrasonic sensors measure distance by using ultrasonic waves. Sensor head emits an ultrasonic wave and receives wave reflected back from target. Ultrasonic sensors measure distance to target by measuring time between emission and reception. Optical sensor has a Tx and Rx, whereas an ultrasonic sensor uses a single ultrasonic element for both emission and reception. In a reflective model ultrasonic sensor, a single oscillator emits and receives ultrasonic waves alternately. This enables miniaturization of the sensor head.



HC-SR04 Ultrasonic Module has 4 pins, Ground, VCC, Trig and Echo. Ground and VCC pins of module needs to be connected to Ground and 5 volts pins on Arduino Board respectively and trig and echo pins to any Digital I/O pin on Arduino Board. To generate ultrasound, set Trig on a High State for 10 us to send out an 8 cycle sonic burst which will travel at speed sound and will be received in Echo pin. Echo pin will output time in microseconds sound wave travelled.

Ultrasonic Sensor Pin out

PIN 1,2,3,4 (from left to right) • **Ground – G** • **12 – trig** • **11 – echo** • **5Volt – Vcc**	

Ultrasonic Sensor Code

```
#define trigPin 11
#define echoPin 12
#define ledPin 13
void setup() {
Serial.begin(9600);
pinMode(trigPin,OUTPUT);
pinMode(echoPin,INPUT);
pinMode(ledPin,OUTPUT);
}
void loop() {
long duration , distance;
digitalWrite(trigPin,LOW);
delayMicroseconds(2);
digitalWrite(trigPin,HIGH);
delayMicroseconds(10);
digitalWrite(trigPin,LOW);
duration = pulseIn(echoPin,HIGH);
distance = (duration/2) / 29.1;
if(distance<10)
{
digitalWrite(ledPin,HIGH);
}else{
digitalWrite(ledPin,LOW);
}
Serial.print(distance);
Serial.println("cm");
delay(1500);
}
```

Result

PASTE YOUR SCREENSHOT OF THE SUCCESSFUL RUN OF PROGRAM INCLUDING HARDWARE CONNECTION AND ARDUINO IDE

Experiment No. 9

Aim: To understand the working and programming of Servo Motor.

Theory

A servo motor is a self-contained electrical device, that rotate parts of a machine with high efficiency and with great precision. Output shaft of this motor can be moved to a particular angle, position and velocity that a regular motor does not have. Servo Motor utilizes a regular motor and couples it with a sensor for positional feedback. Servo motor is a closed-loop mechanism that incorporates positional feedback in order to control the rotational or linear speed and position. Motor is controlled with an electric signal, either analog or digital, which determines the amount of movement which represents the final command position for the shaft.

Servo Motor Pin out

- Servo motors have three wires: power, ground, and signal.
- Power wire is typically red, and connected to 5V pin on Arduino board.
- Ground wire is typically black or brown and connected to ground pin on board.
- Signal pin is typically yellow or orange and should be connected to PWM pin on board.

Servo Motor Code 1

```
#include <Servo.h>
Servo myservo; // create servo object to control a servo
int pos = 0; // variable to store the servo position
void setup() {
myservo.attach(9); // attaches the servo on pin 9 to the servo object
}

void loop() {
for (pos = 0; pos <= 180; pos += 1)
{
myservo.write(pos); // tell servo to go to position in variable 'pos'
delay(15); // waits 15ms for the servo to reach the position
}
for (pos = 180; pos >= 0; pos -= 1)
{
myservo.write(pos); // tell servo to go to position in variable 'pos'
delay(15); // waits 15ms for the servo to reach the position
```

```
}
}
```

Servo Motor Code 2

```
#include <Servo.h>
Servo myservo;  // Create servo object

void setup() {
  Serial.begin(9600);  // Initialize serial communication
  myservo.attach(9);    // Attach servo to pin 9
}

void loop() {
  Serial.print("Enter desired angle (0-180): ");
  int val = Serial.parseInt();
  if (val >= 0 && val <= 180) {
    Serial.print("Setting servo to: ");
    Serial.println(val);
    myservo.write(val);  // Set servo position
  } else {
    Serial.println("Invalid angle. Please enter a value between 0 and 180.");
  }
  delay(1000);
}
```

Result

PASTE YOUR SCREENSHOT OF THE SUCCESSFUL RUN OF PROGRAM INCLUDING HARDWARE CONNECTION AND ARDUINO IDE

Experiment No. 10

Aim: Adding an LCD screen and sketch walkthrough

Theory

The LiquidCrystal library allows you to control LCD displays that are compatible with the Hitachi HD44780 driver. There are many of them out there, and you can usually tell them by the 16-pin interface. The LCDs have a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display.

The interface consists of the following pins:

A register select (RS) pin that controls where in the LCD's memory you're writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.

A Read/Write (R/W) pin that selects reading mode or writing mode

An Enable pin that enables writing to the registers

8 data pins (D0 -D7). The states of these pins (high or low) are the bits that you're writing to a register when you write, or the values you're reading when you read.

There's also a display contrast pin (Vo), power supply pins (+5V and GND) and LED Backlight (Bklt+ and BKlt-) pins that you can use to power the LCD, control the display contrast, and turn on and off the LED backlight, respectively.

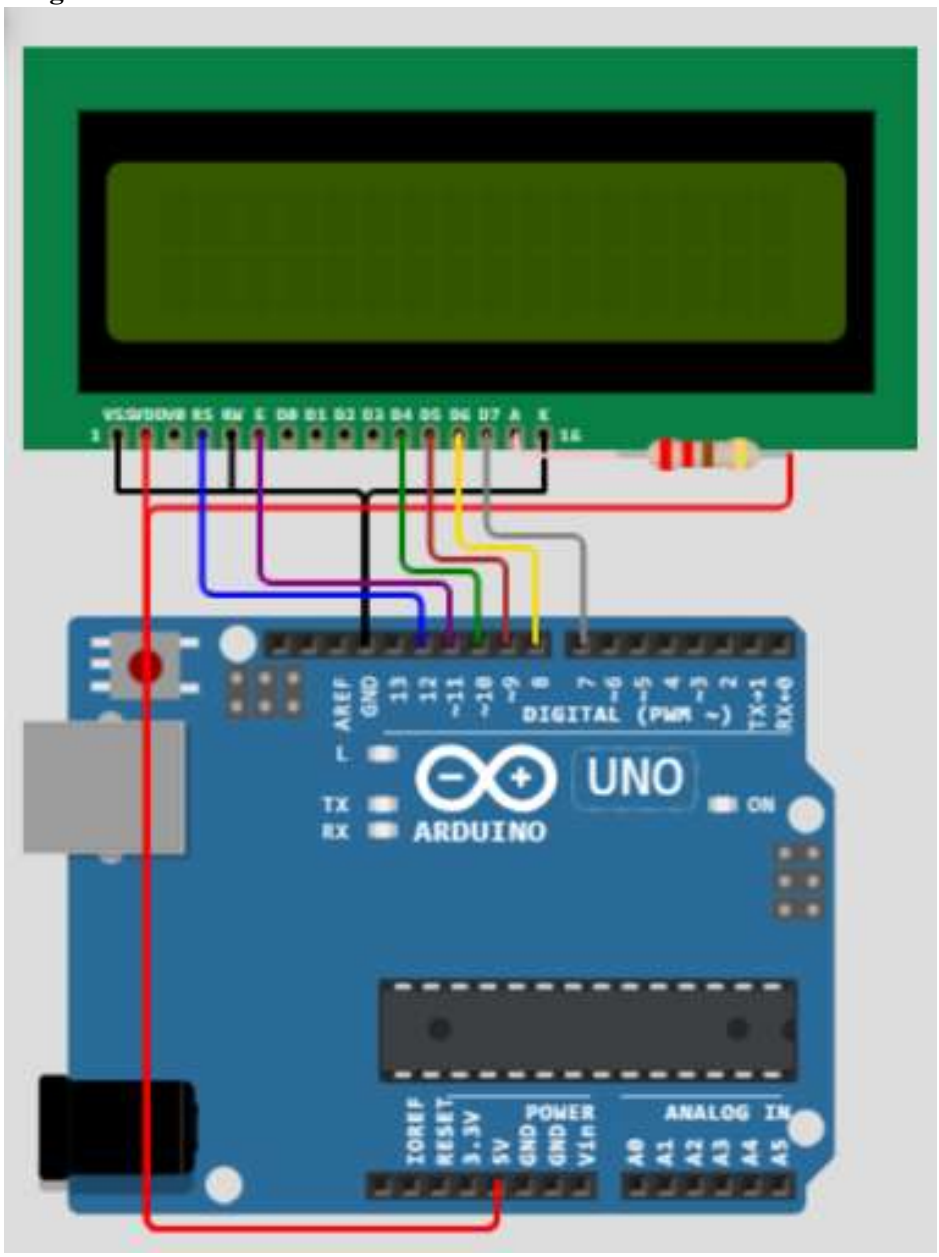
The process of controlling the display involves putting the data that form the image of what you want to display into the data registers, then putting instructions in the instruction register. The LiquidCrystal Library simplifies this for you so you don't need to know the low-level instructions. Hitachi-compatible LCDs can be controlled in two modes: 4-bit or 8-bit. The 4-bit mode requires seven I/O pins from the Arduino, while the 8-bit mode requires 11 pins. For displaying text on the screen, you can do most everything in 4-bit mode, so example shows how to control a 16x2 LCD in 4-bit mode.

Arduino and LCD Pin out

Name	Description	Arduino Pin*
VSS	Ground	GND.1
VDD	Supply voltage	5V
V0	Contrast adjustment (not simulated)	
RS	Command/Data select	12
RW	Read/Write. Connect to Ground.	GND.1

E	Enable	11
D0 – D3	Parallel data 0 - 3 (optional) †	
D4	Parallel data 4	10
D5	Parallel data 5	9
D6	Parallel data 6	8
D7	Parallel data 7	7
A	Backlight anode	5V / 6‡
K	Backlight cathode	GND.1

Wiring Diagram

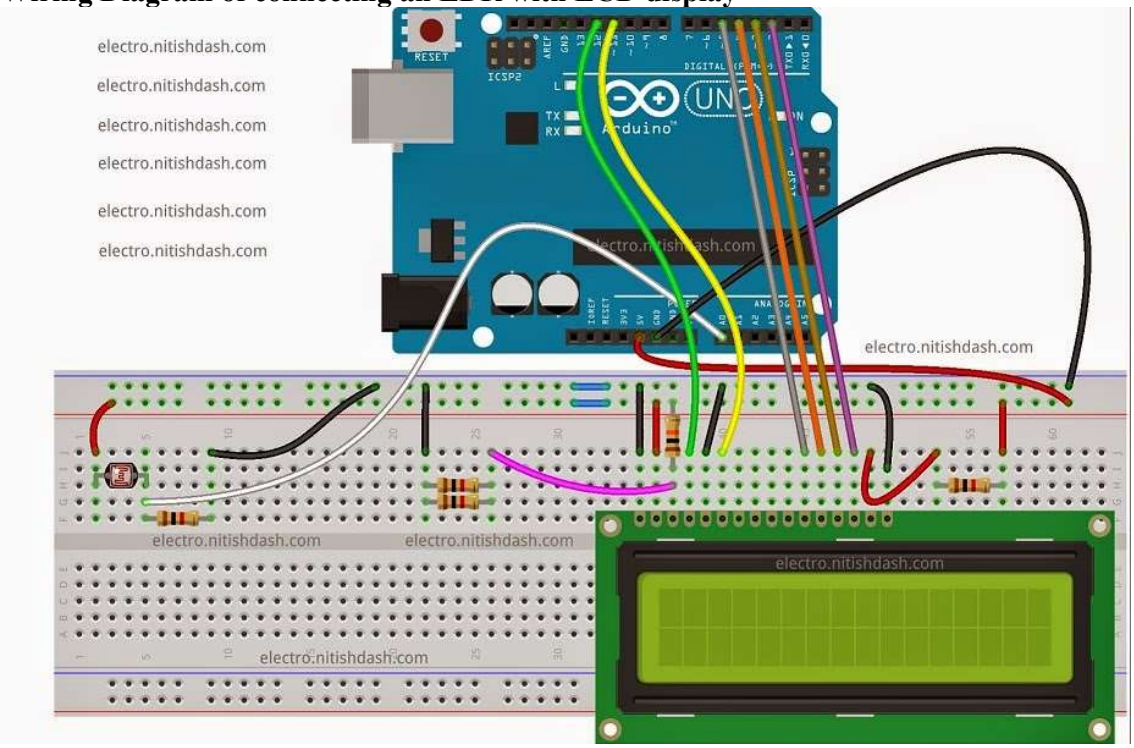


Simple Hello Program

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 10, 9, 8, 7);
void setup() {
  lcd.begin(16, 2);
  // you can now interact with the LCD, e.g.:
  lcd.print("Hello World!");
}

void loop() {
  // ...
}
```

Wiring Diagram of connecting an LDR with LCD display



Code

```
/*  The circuit:
 * LCD RS pin to digital pin 12
 * LCD Enable pin to digital pin 11
 * LCD D4 pin to digital pin 10
 * LCD D5 pin to digital pin 9
 * LCD D6 pin to digital pin 8
 * LCD D7 pin to digital pin 7
 * LCD R/W pin to ground
 * LCD VSS pin to ground
 * LCD VCC pin to 5V
 * 10K resistor:  * ends to +5V and ground  * wiper to LCD VO pin (pin 3)
*/

 #include <LiquidCrystal.h>
LiquidCrystal lcd(7, 8, 9, 10, 11 , 12);
```

```
void setup() {
  lcd.begin(16, 2);
  lcd.setCursor(0,1);
  lcd.write("LIGHT: ");
}
void loop()
{
  int sensorValue = analogRead(A0);
  lcd.print("Room: ");
  if (sensorValue > 700)
  {
    lcd.print("Light!");
  }
  else
  {
    lcd.print("Dark  ");
  }
  delay(100);
}
```

Result

PASTE YOUR SCREENSHOT OF THE SUCCESSFUL RUN OF PROGRAM INCLUDING HARDWARE CONNECTION AND ARDUINO IDE