Lab 4: SENSOR INTERFACES

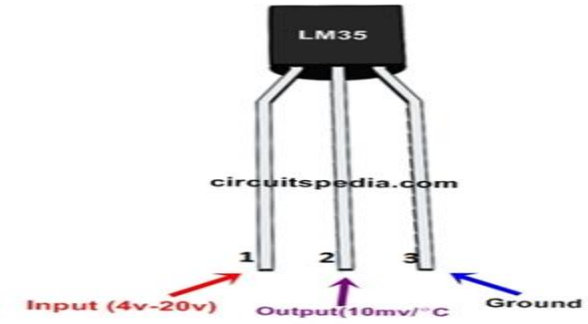
1. **Objectives: After this lab, the learner will be able to:**
   - Interface sensors module with Arduino board
2. **Facilities**
   - Proteus software
   - Arduino IDE
3. **Prerequest**
   - Basic electronic
   - C programming
4. **Notes: Before do this lab, do followings:**
   - Download file: All-in-1-Arduino-Sensor-Libraries-for-Proteus.Zip
   - Extract file and then copy all file in All-in-1-Arduino-Sensor-Libraries-for-Proteus folder and paste into Library folder with following path:
   - *Proteus 8 Users*
     C:\Documents and Settings\All Users\Application Data\Labcenter Electronics\ Proteus 8 Professional\LIBRARY
   - *Proteus 7 Users*
     C:\Labcenter Electronics\Proteus 7 Professional\LIBRARY
   - Notes: if you use windows-10, path of following is used:
     C:\ProgramData\Labcenter Electronics\Proteus 8 Professional\LIBRARY
5. **Interface with temperature sensor LM35**

5.1. Introduction

Thermometers enable us to read the temperature in any room, space or region. Thermometers are widely used in industries for temperature monitoring during the manufacturing process. The need of analog thermometers is decreasing with the passage of time as analog thermometers cannot be used for remote monitoring of temperature, they are slow and obsolete. However, digital thermometers are developed with microcontrollers nowadays and offer various features which make them more practical and useful. For example, digital thermometers can be used in remote areas where the physical presence of an operator is impractical such as in hot furnace at an industry. The thermometer reads temperature and wirelessly transmits signals to the mobile or HMI system in the monitoring room. Furthermore, the temperature range can be set to Celsius or Fahrenheit according to requirement and can be displayed on a screen or in the form of LED display.

5.2. Theory

LM35 which is a temperature sensor and looks more like a simple BJT. LM35 is cheap as compared to most of the temperature sensors and yet offers a high level of accuracy even at extreme temperatures. LM35 can be used in both analog circuits and embedded systems since it offers analog voltages at the output.



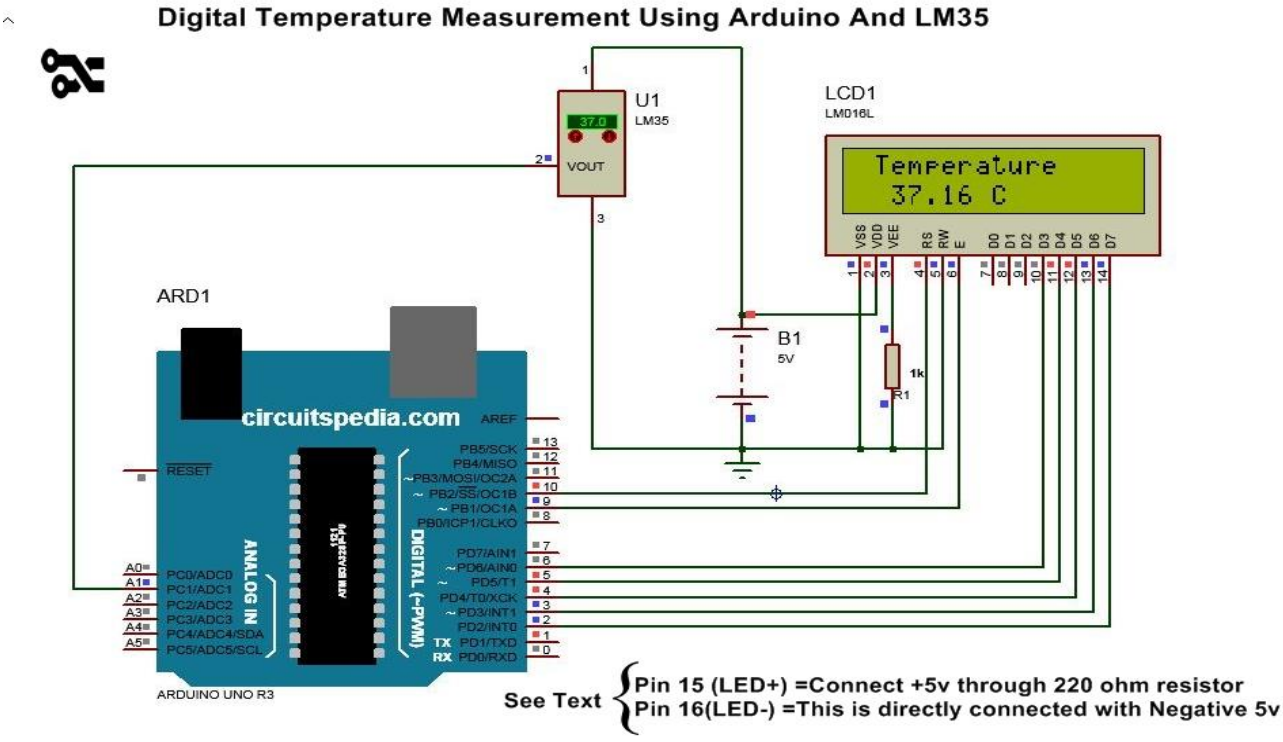
Calculation of temperature is not enough, to display our results, **HDD44780** driver based **16×2 LCD** has been used. 16×2 stands for 16 rows and 2 columns incorporated in this design. This model is widely used for simple display as it is low cost and works efficiently. The LCD is compatible with Arduino as the protocols involved are satisfied at both ends. The LCD is available in 16×4 configuration too in case if more rows are needed.

5.3. Components required

The components required for this design are:

- Arduino UNO MCU
- LM35 temperature sensor
- 16×2 LCD (HDD44780 driver based)
- 6v battery or 5v USB power source (Mobile adapter/laptop)
- Jumper wires
- Breadboard

5.4. Schematic diagram



Pin 15 and Pin 16 Of LCD is for Backlight . If these not connected then only backlight will remain off. Connect the pin 15 of LCD with +ve 5v . This is the positive terminal of Backlight LED. Pin 16 is negative terminal of backlight LED, so pin 16 is Grounded.

5.5. Simulation

The circuit diagram described in the figure is simple and can be easily modified if needed. Whenever designing a circuit, it is recommended first to simulate and check if the results are according to requirements. Since simulation can be easily modified with a few clicks but hardware is difficult to modify as the soldering, unsoldering, troubleshooting can be hectic and time-consuming.

LM35 offers analog voltages with respect to the temperature it is kept in. Since the temperature cannot be changed in simulation, the simulated model can be controlled using the temperature buttons. As the temperature for a simulation model of LM35 is variated, the instructions for rising temperature are forwarded to the sensor. LM35 offers rise of 10mv/degree Celsius.

As these output voltages of LM35 are analog and analog values cannot be directly processed by the MCU, these values are first passed to Analog to Digital Converter. The digitally converted value is further processed as per the algorithm. These digital inputs are used for calculation of temperature in degree Celsius and Fahrenheit.

The temperature is further displayed on the LCD connected with Arduino. The LCD has data pins, read/write enable configurations for further features.

It is essential to connect each connection as demonstrated in simulation. The battery source voltage must be maintained 5v. Although in simulation, even if voltages exceed 5v, they cause no harm to the components but practically, the rated voltages of components must be considered. Exceeding 5v can permanently damage components.

After making all necessary connection, ensure each connection and value of the component is according to the design. Add hex file of code in Arduino UNO's simulation model. The hex file serves as the machine language instruction for the MCU.

5.6. Coding

```
// Digital Thermometer
#include <LiquidCrystal.h>
LiquidCrystal LCD(10, 9, 5, 4, 3, 2);
```

```
        float tempC;
        int reading;
        int tempPin = 1;
        void setup()
        {
        LCD.begin(16,2);
        LCD.setCursor(0,1);
        LCD.print("    ");
        LCD.setCursor(0,0);
        LCD.print("    ");
        analogReference(INTERNAL);
        Serial.begin(9600);
        }
void loop()
{
        reading = analogRead(tempPin);
        tempC =  reading / 9.31;
        Serial.println(tempC);
        LCD.setCursor(0,0);
        LCD.print(" Temperature ");
        LCD.setCursor(0,1);
        LCD.print("  ");
        LCD.print(tempC);
        LCD.print(" C  ");
        delay(200);
        LCD.setCursor(0,1);
        LCD.print("   ");
}
```

6. Interfacing with DHT11 sensor

6.1. Introduction

This topic shows how to interface Arduino UNO board with DHT11 digital humidity and temperature sensor where the measure humidity and temperature are displayed on 1602 LCD screen.

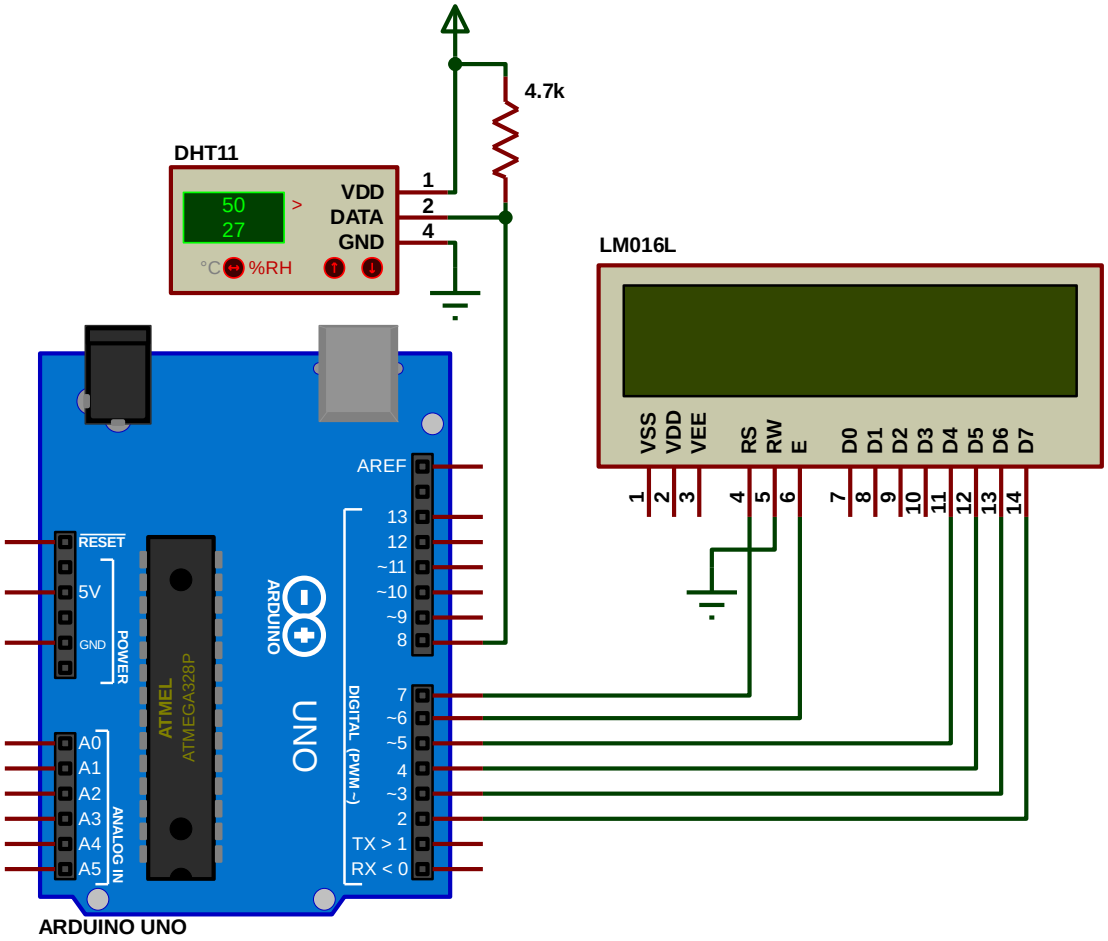
6.2. DHT11 Sensor technical specifications

- Humidity Range: 20-90% RH
- Humidity Accuracy: ±5% RH
- Temperature Range: 0-50 °C
- Temperature Accuracy: ±2 °C
- Operating Voltage: 3V to 5.5V
- Resolution = 1

6.3. Hardware Required:

- Arduino board
- DHT11 (RHT01) sensor
- 1602 LCD screen
- 4.7K ohm resistor
- 10K ohm variable resistor
- 330 ohm resistor
- Breadboard
- Jumper wires

6.4. Arduino + DHT11 + LCD connection circuit:



As show in the circuit schematic the DHT11 sensor has 4 pins: VCC (+5V), Data, NC(not connected pin) and GND (from left to right). A pull-up resistor should be added to the data pin with a value between 4.7K and 10K.

The 10K variable resistor is used to control the contrast of the LCD screen and the 330 ohm resistor supplies the LCD backlight LED.

6.5. Arduino code:

In this interfacing I used DHT sensors library (DHT11, DHT21 and DHT22) from Adafruit. This library initiates the DHT11 sensor and reads the values of the humidity and temperature.

Since the DHT11 sensor resolution is 1, the values of the humidity and temperature are stored in two variables with type byte (8-bit unsigned), I named them RH (for the humidity) and Temp (for the temperature).

The values of the humidity and temperature needs to be displayed on the LCD after the reading, and for that I used two character arrays named: temperature (char temperature[] = "Temp = 00.0 C ";) and humidity (char humidity[] = "RH = 00.0 % ";). Humidity and temperature values are copied to the previous arrays before displaying it. For example the temperature value is copied to the array temperature as follows:

```
temperature[7] = Temp / 10 + 48;
```

This line means the 7th character of the array is equal to the tens of temperature value

```
temperature[8] = Temp % 10 + 48;
```

(Temp%10 equals to the remainder of Temp/10)
and this line means the 8th character of the array is equal to the ones of temperature value. For example if the temperature value is 42 ==> 42 / 10 = 4 and 42 % 10 = 2.

The number 48 is used to convert decimal numbers to AscII because the LCD works with AscII format.

The line `temperature[11] = 223;` means adding the degree symbol (°).

6.6. Code

```
// Interfacing Arduino with DHT11 humidity and temperature sensor

 // include LCD library code

#include <LiquidCrystal.h>

// include DHT library code

#include "DHT.h"

 #define DHTPIN 8          // DHT11 data pin is connected to Arduino pin 8

 // LCD module connections (RS, E, D4, D5, D6, D7)

LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
```

```
 #define DHTTYPE DHT11       // DHT11 sensor is used
DHT dht(DHTPIN, DHTTYPE);   // Initialize DHT library
 char temperature[] = "Temp = 00.0 C  ";
char humidity[]    = "RH   = 00.0 %  ";
void setup() {
  // set up the LCD's number of columns and rows
  lcd.begin(16, 2);
  dht.begin();
}
 void loop() {
  delay(1000);          // wait 1s between readings
  // Read humidity
  byte RH = dht.readHumidity();
  //Read temperature in degree Celsius
  byte Temp = dht.readTemperature();
   // Check if any reads failed and exit early (to try again)
  if (isnan(RH) || isnan(Temp)) {
    lcd.clear();
    lcd.setCursor(5, 0);
    lcd.print("Error");
    return;
  }
   temperature[7]     = Temp / 10 + 48;
  temperature[8]     = Temp % 10 + 48;
  temperature[11]    = 223;
  humidity[7]        = RH / 10 + 48;
  humidity[8]        = RH % 10 + 48;
  lcd.setCursor(0, 0);
  lcd.print(temperature);
```

```
  lcd.setCursor(0, 1);
  lcd.print(humidity);
}
```

7. Interface with Ultrasonic Sensor

7.1. Introduction

Ultrasonic sensors not only offer distance measuring utility without any physical contact but enable us to measure with no noise and light, unlike laser-based distance measuring instruments. Furthermore, these instruments are cheap and more reliable even in daylight where laser-based instruments often decrease their efficiency.Distance measuring instruments have been in use for centuries and improvements have been made in their designs with the passage of time. Today, Distance measuring instruments such as foot ruler and inches tape are obsolete and digital instruments are used for such purposes at a larger scale. High precision and more convenience of measuring any distance from one point have made the process easy. Such instruments are widely used at construction sites and fluid level monitoring. In containers and sites where fluid level monitoring needs to be precise and remotely controlled, ultrasonic sensors based distance measuring instruments provide ease. Since the design is based on embedded systems and the whole process is controlled by a microcontroller, many features can be added to it. For example, remotely transmission of fluid level and accordingly ON/OFF feature for a digital fluid switch.

7.2. Theory

Ultrasonic sensors also known as Sonar sensors have been in use for decades. They were once used for mapping of ships in sea and detection of broken/faulty parts in mechanical systems. The ultrasonic sensor works by transmitting ultrasonic waves and these waves strike with obstacles placed in front of the transmitter. The waves are reflected and hit on the receiver. The time taken by the waves from transmission to receiving with respect to the velocity of ultrasonic waves is used to measure the distance of obstacle present ahead.

The speed of sound is approximately 341 meters (1100 feet) per second in air. The ultrasonic distance sensor uses this information along with the time difference between sending and receiving the sound signal to determine the distance between object. It uses the following mathematical formula.

Distance = Time x Speed of Sound divided by 2

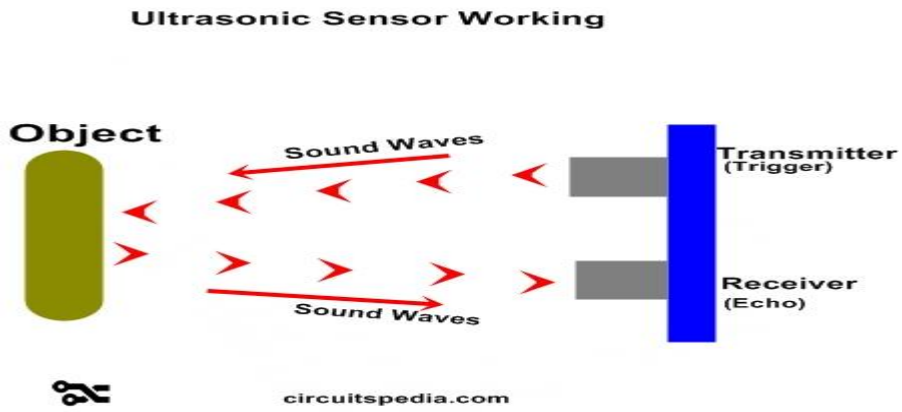
Time = the time between when an ultrasonic wave is transmitted and when it is received

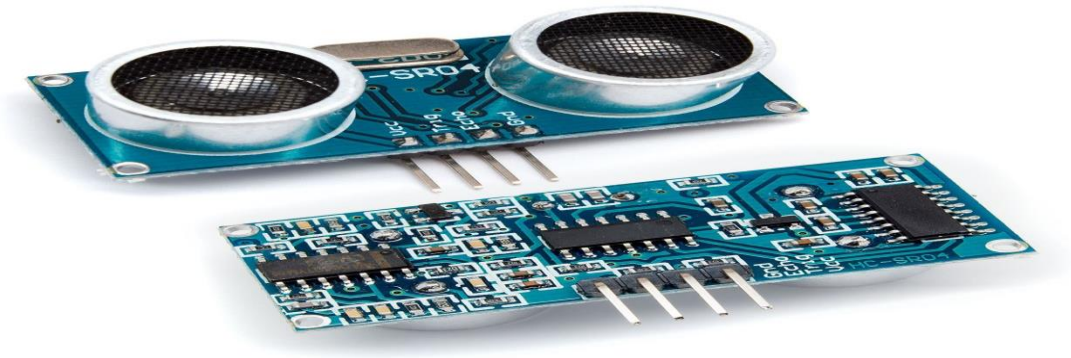
this number is divided by 2 because the sound wave has to travel to the object and back.

Since the distance can't be changed in physically in simulation, the simulation model of **SR04** is connected with a potentiometer, and the distance for the sensor is varied using this potentiometer.

7.3. Ultrasonic Sensor Range

HC SR-04 Ultrasonic Distance Sensor Provides The Range between 2cm to 400 cm approx (1 inch to 13 foot). With Fully non contact .





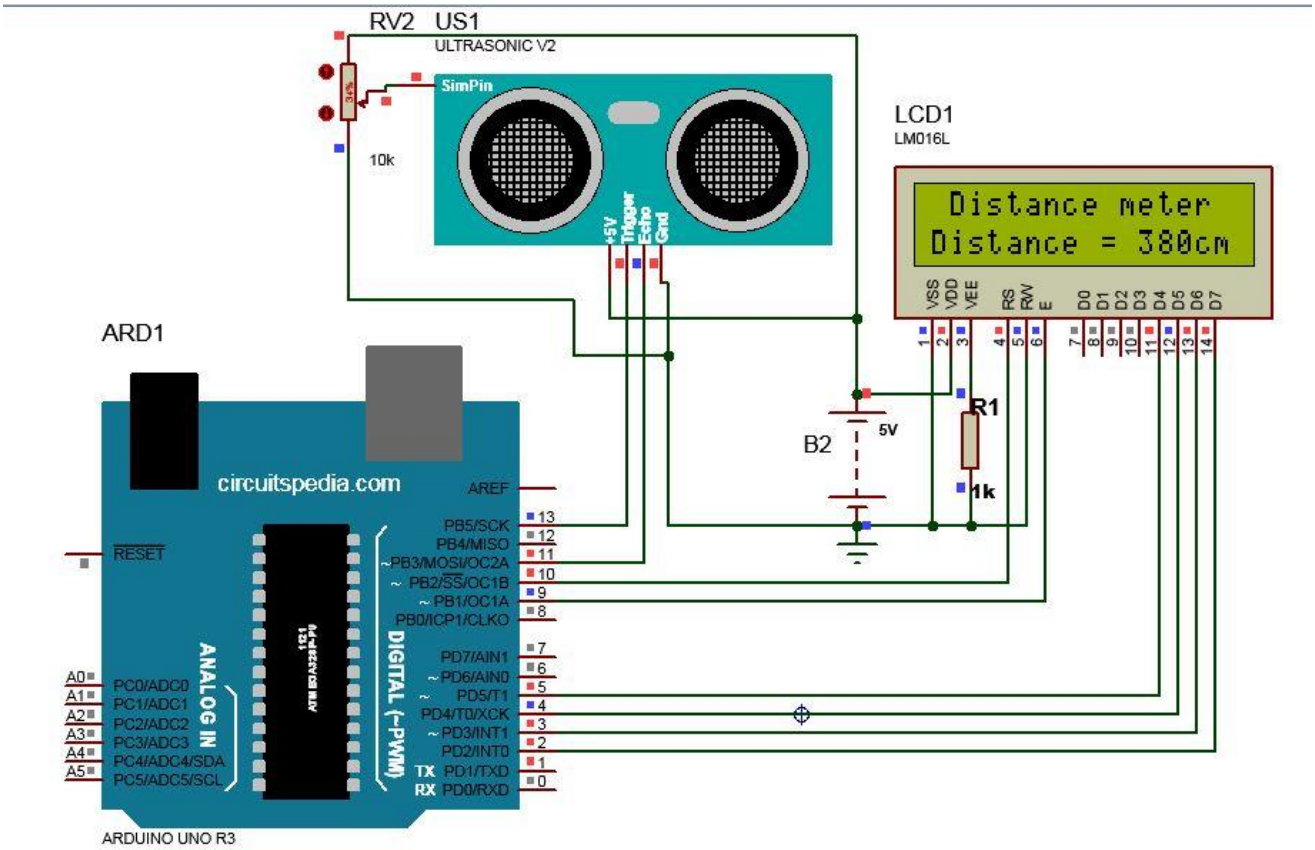
Measurement of distance needs to be expressed in some way, and for embedded systems, the best way is to use an LCD. 16×2 HDD44780 driver based LCD offers efficient and smooth functioning at a cheap price. The 16 columns and 2 rows offer enough space to display results. The LCD is compatible with all models of Arduino boards which makes it suitable for this project too.

7.4. Components required

- The components required for this project are simple and low cost.
- Arduino UNO MCU
- Ultrasonic sensor SR04
- 16×2 LCD (HDD44780 driver based)
- 6v battery or 5v USB power source (Mobile adapter/laptop)

- Potentiometer 10k ohm
- Jumper wires
- Breadboard

7.5. Connection Diagram



The working mechanism of this project is simple and easy to understand. The ultrasonic sensor has 4 pins out of which 2 pins are for power, and the remaining 2 pins are for the trigger and echo. The MCU transmits ultrasonic waves through the trigger pin and then reads the receiving time of that ultrasonic wave on echo pin. The velocity of the ultrasonic wave traveling is defined and accordingly the distance is measured.

It is essential to add a hex file of Arduino code in its simulation model. The hex file is in the form of binary instructions given to the MCU for processing.

LCD serves as a graphical interface for displaying results. The display brightness can be adjusted by supplying 0-5v on pin 3 which is unnecessary in case of simulation but works well for hardware design. Make sure the pins are carefully connected with Arduino as mentioned in the circuit diagram.

7.6. Working

The use of 10k Preset is only in simulation for making a Object between sensor . Here the preset is works as object.

The use of a potentiometer with the Ultrasonic sensor enables the user to control the input distance for sensor and observe changes in results accordingly. The value of this potentiometer must be above 1K ohm.

Since the simulation model ensures proper functioning of the design, if your simulation works correctly, you can proceed towards the development of hardware design. However in case if your simulation is not working correctly, you need to troubleshoot your error.

If your LCD is displaying distance and it varies but the distance is incorrect, you might be directing your Ultrasonic sensor in the wrong manner, make sure the path is clear between target obstacle and Ultrasonic sensor and the sensor is facing an obstacle. Alternatively, the connections of the ultrasonic sensor with Arduino might be wrong. If there is no display on the LCD, then you need to check code and connections of LCD with Arduino.

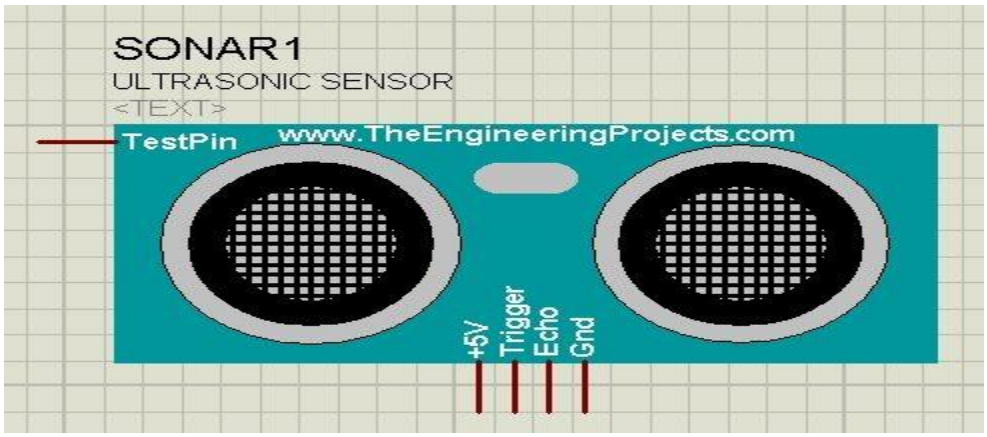
Failing to connect all components as instructed in the circuit diagram, the simulation might not work or show unexpected behavior.

7.7. Coding

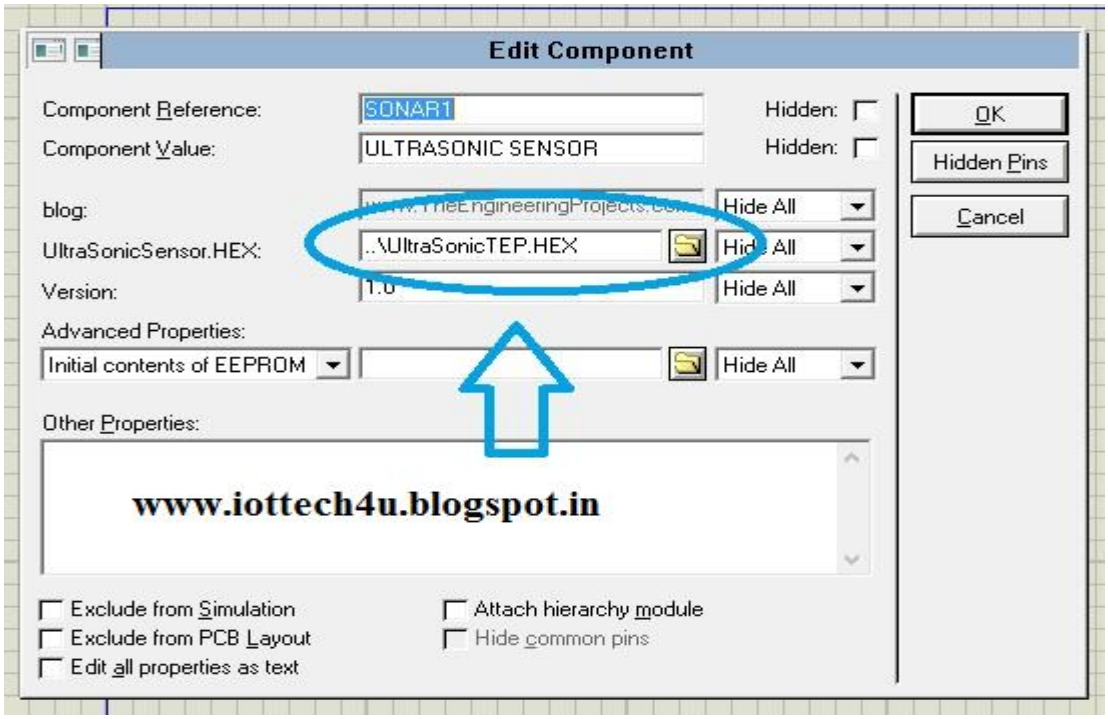
```
#include <LiquidCrystal.h>;

LiquidCrystal lcd(10,9,5,4,3,2);

const int trigPin=13;  // defines the pin numbers

const int echoPin=11;

long duration;                // defines variables

int distanceCm, distanceInch;

void setup()

Serial.begin (9600)

lcd.begin (16,2);// Initializes the interface to the LCD screen, and specifies the dimensions (width and height) of the display

pinMode (trigPin, OUTPUT);

pinMode (echoPin, INPUT);

}
```

```
void loop() {

      digitalWrite(13,LOW);  // Clears the trigPin

      delayMicroseconds(2);

      digitalWrite(13,HIGH); // Sets the trigPin on HIGH state
for 10 micro seconds

      delayMicroseconds(10); // Sets the trigPin on HIGH state
for 10 micro seconds

      digitalWrite(13,LOW);

      duration = pulseIn(11, HIGH);//To receive the reflected
signal.

      distanceCm= duration*0.0340/2;  // Calculating the dista
nce

      distanceInch = duration*0.0133/2;

      lcd.setCursor(0,0); // Sets the location where text writ
ten to the LCD will be displayed

      lcd.print("Distance= "); // Prints string "Distance" on
the LCD

      lcd.print(distanceCm);   // Prints the distance value fr
om the sensor

      lcd.print(" cm    ");

      delay(20);

      lcd.setCursor(0,1);

      lcd.print("Distance= ");

      lcd.print(distanceInch);

      lcd.print(" inch    ");

      delay(20);

}
```

Notes: in case of ultrasonic sensor dosnt't working – cant'n open Hexfile (Now we have our ultrasonic sensor in Proteus but if you run it then it won't work as we haven't yet added any functionality in it.)



- So, in order to add the functionality double click this ultrasonic sensor and open its properties.
- In properties, select the Program File section and browse to UltrasonicTEP.HEX file and upload it as shown in below figure:



- Now our ultrasonic sensor is ready to be used.
- Now let's make a simple example for the ultrasonic sensor so that you get an idea how to use it in Proteus.

8.