PRACTICAL 7

AIM: LCD will be used with Arduino / ESP32 with various sensors.

PREREQUISITE: Basics of programming, microcontrollers and basic electronics.

OUTCOME: 1. Study and work of LCD

2. Connecting microcontroller board with LCD
3. Display of sensor data over the 16X2 LCD.

PART 1

THEORY:

1. **Study and Work of LCD:**

   An LCD (Liquid Crystal Display) is a flat-panel display that uses liquid crystals to produce visible images. It works by modulating the light through liquid crystals sandwiched between polarized glass. LCDs are commonly used in electronic devices for displaying text, numbers, and graphics. A typical 16x2 LCD has 16 columns and 2 rows, allowing it to display up to 32 characters simultaneously.

2. **Connection of 16x2 LCD with Arduino and ESP32 Microcontroller Boards:**

   To connect a 16x2 LCD with Arduino or ESP32, the LCD requires connections to data pins (D4-D7 for 4-bit mode), RS (Register Select), E (Enable), and VSS/VDD for power. Using libraries like LiquidCrystal in Arduino and similar libraries for ESP32, you can easily control the LCD and send characters or messages to display. Both microcontrollers can interface with the LCD through digital pins and use serial communication to control the display.

3. **Display the Values Sensed by Various Sensors:**

   The 16x2 LCD can display sensor data such as distance from an ultrasonic sensor, light intensity from a photosensitive resistor, or resistance values from a potentiometer. The sensor readings are processed by the microcontroller (Arduino or ESP32), which converts analog/digital sensor values into readable text, and then sends this data to the LCD for real-time display.

4. **Use a DHT Sensor to Display the Temperature and Humidity:**

   A DHT sensor (like DHT11 or DHT22) is used to measure temperature and humidity. It is connected to a microcontroller like Arduino or ESP32, and the sensor values are fetched using appropriate libraries. These values are then displayed on the 16x2 LCD. The DHT sensor outputs digital values, which are easily interpreted and shown on the screen in a human-readable format (e.g., "Temp: 25°C, Humidity: 60%").

PART 2

Ultrasonic Sensor and LCD:

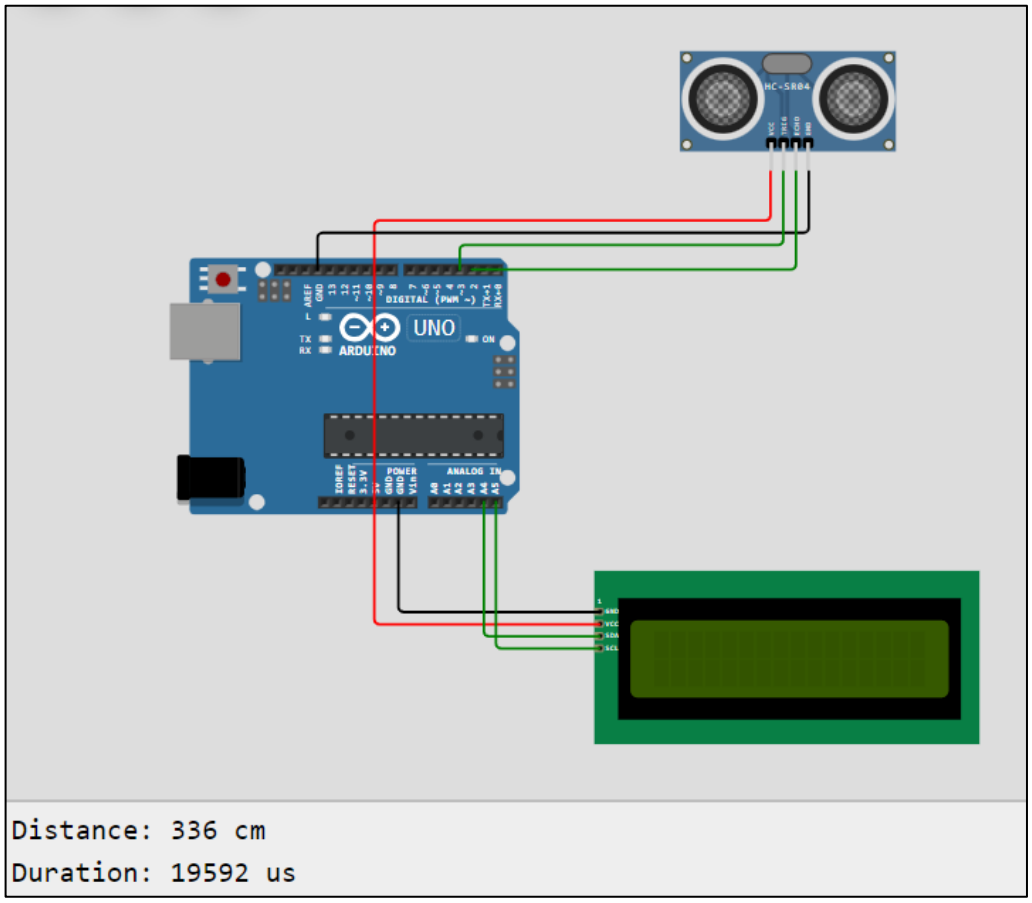
CODE:

```
#include <Wire.h>                 // Library for I2C communication
#include <LiquidCrystal_I2C.h>    // Library for I2C LCD

// Initialize the I2C LCD
LiquidCrystal_I2C lcd(0x27, 16, 2);   // Set the LCD address to 0x27 (may vary) for a 16 chars and 2 line display

// Define Ultrasonic Sensor Pins
const int trigPin = 3;
const int echoPin = 2;

long duration;
int distance;

void setup() {
  // Set up the ultrasonic sensor pins
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
```

```
  // Initialize LCD
  lcd.init();
  lcd.backlight();

  // Begin Serial Communication for debugging
  Serial.begin(9600);
}

void loop() {
  // Clear the trigger pin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  // Send a 10us pulse to trigger the ultrasonic sensor
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Read the echo pin, and calculate the duration in microseconds
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance in centimeters
  // Speed of sound is 343 m/s or 0.0343 cm/us
  distance = duration * 0.0343 / 2;

  // Print distance and duration on the LCD
  lcd.clear();
  lcd.setCursor(0, 0);   // Set cursor to the first line
  lcd.print("Dist: ");
  lcd.print(distance);
```

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

  lcd.setCursor(0, 1);   // Set cursor to the second line
  lcd.print("Dur: ");
  lcd.print(duration);
  lcd.print(" us");

  delay(3000);  // Delay before next reading
}
```

OUTPUT:



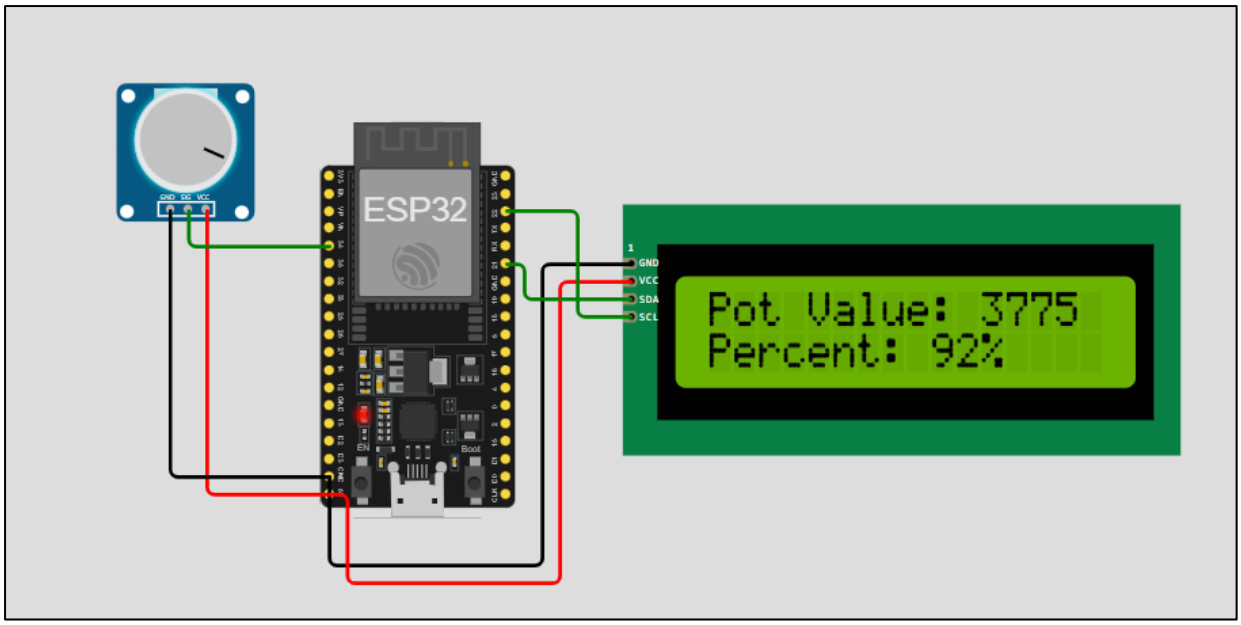
ESP32 and Potentiometer:

CODE:

```
#include <Wire.h>                 // Library for I2C communication
#include <LiquidCrystal_I2C.h>    // Library for I2C LCD

// Initialize the I2C LCD
LiquidCrystal_I2C lcd(0x27, 16, 2);   // Set the LCD address to 0x27 (adjust if needed)

// Define the pin for the potentiometer
#define POT_PIN 34                // Potentiometer connected to GPIO 34 (analog input on ESP32)

void setup() {
  // Initialize LCD
  lcd.init();
  lcd.backlight();

  // Display a welcome message on the LCD
  lcd.setCursor(0, 0);
  lcd.print("Potentiometer");
  lcd.setCursor(0, 1);
  lcd.print("Value:");
  delay(2000);                    // Wait 2 seconds before starting readings
  lcd.clear();
}

void loop() {
  // Read the potentiometer value (0 to 4095 for ESP32's 12-bit ADC)
  int potValue = analogRead(POT_PIN);
```

```
  // Map the potentiometer value to a percentage (0 to 100)
  int potPercent = map(potValue, 0, 4095, 0, 100);

  // Print potentiometer value on the LCD
  lcd.setCursor(0, 0);            // Set cursor to the first line
  lcd.print("Pot Value: ");
  lcd.print(potValue);

  lcd.setCursor(0, 1);            // Set cursor to the second line
  lcd.print("Percent: ");
  lcd.print(potPercent);
  lcd.print("%");

  delay(500);                     // Wait for 0.5 seconds before taking the next reading
}
```

OUTPUT:



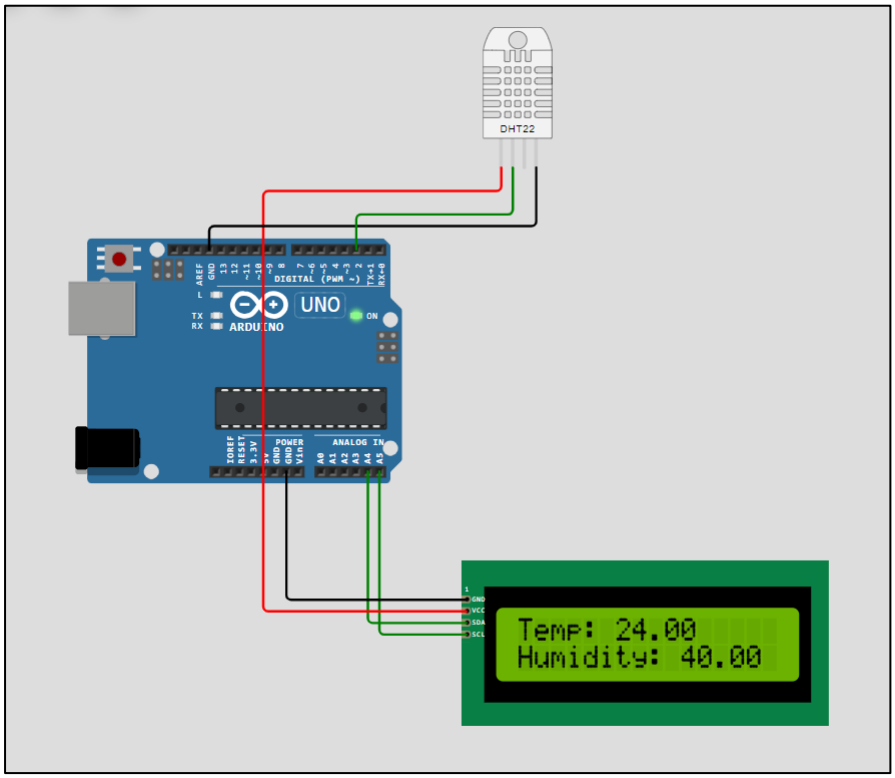
DHT and Arduino:

CODE:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

#define DHTPIN 2
#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  lcd.init();
  lcd.backlight();

  dht.begin();

  lcd.setCursor(0, 0);
  lcd.print("Temp & Humidity");
  delay(2000);
  lcd.clear();
}

void loop() {
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();
  if (isnan(humidity) || isnan(temperature)) {
```

```
    lcd.setCursor(0, 0);
    lcd.print("Sensor Error!");
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(temperature);

  lcd.setCursor(0, 1);
  lcd.print("Humidity: ");
  lcd.print(humidity);
  delay(2000);
}
```

OUTPUT:



Observation:

While working with a 16x2 LCD connected to Arduino and ESP32, it was observed that the microcontrollers efficiently interface with the display to show real-time data from various sensors like ultrasonic, photosensitive resistors, and potentiometers. The LCD provided a clear and responsive visual output of sensor values. The DHT sensor accurately measured and displayed the temperature and humidity on the LCD. Both Arduino and ESP32 platforms successfully handled multiple inputs and output tasks, making them ideal for IoT applications requiring real-time monitoring.

Conclusion:

The integration of a 16x2 LCD with Arduino and ESP32 demonstrates how IoT systems can effectively communicate sensor data to users in a simple visual format. The ability to display real-time sensor readings, such as environmental conditions (temperature, humidity) or proximity, is vital for many practical applications, including home automation, weather stations, and smart systems. This combination of microcontrollers, sensors, and display units offers a powerful yet accessible tool for prototyping and deploying IoT projects.