AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Faculty of Engineering

Lab Report

Experiment # 07

Experiment Title: Interfacing the Arduino with an external sensor using serial communication protocol for implementing an obstacle detection system.

Date of Perform:	5 May 2025	**Date of Submission:**	12 May 2025
Course Title:	Microprocessor and Embedded Systems Lab		
Course Code:	EE4103	**Section:**	P
Semester:	Spring 2024-25	**Degree Program:**	BSc in CSE
Course Teacher:	**Prof. Dr. Engr. Muhibul Haque Bhuyan**		

Declaration and Statement of Authorship:

1. I/we hold a copy of this Assignment/Case Study, which can be produced if the original is lost/damaged.
2. This Assignment/Case Study is my/our original work; no part has been copied from any other student's work or any other source except where due acknowledgment is made.
3. No part of this Assignment/Case Study has been written for me/us by any other person except where such collaboration has been authorized. by the concerned teacher and is acknowledged in the assignment.
4. I/we have not previously submitted or am submitting this work for any other course/unit.
5. This work may be reproduced, communicated, compared, and archived to detect plagiarism.
6. I/we permit a copy of my/our marked work to be retained by the Faculty Member for review by any internal/external examiners.
7. I/we understand that Plagiarism is the presentation of the work, idea, or creation of another person as though it is your own. It is a form of cheating and is a very serious academic offense that may lead to expulsion from the University. Plagiarized material can be drawn from, and presented in, written, graphic, and visual forms, including electronic data, and oral presentations. Plagiarism occurs when the origin of the source is not appropriately cited.
8. I/we also understand that enabling plagiarism is the act of assisting or allowing another person to plagiarize or copy my/our work.

** Student(s) must complete all details except the faculty use part.*

** Please submit all assignments to your course teacher or the office of the concerned teacher.

Group # 01

Sl No	Name	ID	PROGRAM	SIGNATURE
1	Md. Saikot Hossain	23-51242-1	BSc in CSE	
2	Md. Mosharof Hossain Khan	23-51259-1	BSc in CSE	
3	Rimal Banik	23-51260-1	BSc in CSE	
4	Md. Rahidul Islam	23-51269-1	BSc in CSE	
5	Rahat Ahmed	21-44911-2	BSc in CSE	

Faculty use only

FACULTY COMMENTS	**Marks Obtained**	
	Total Marks	

Contents

Objectives:

The objectives of this experiment are to-

a) Write code for a simple obstacle detection system in Arduino IDE.
b) Implement a simple obstacle detection system using an Arduino microcontroller.

Apparatus:

1. Arduino IDE 2.3.5
2. Arduino UNO Microcontroller board
3. Sonar Sensor (HCSR04)
4. LED lights (RED, GREEN, and YELLOW)
5. Three 100 ohms resistors
6. Jumper wires and
7. Breadboard

Circuit Diagram:

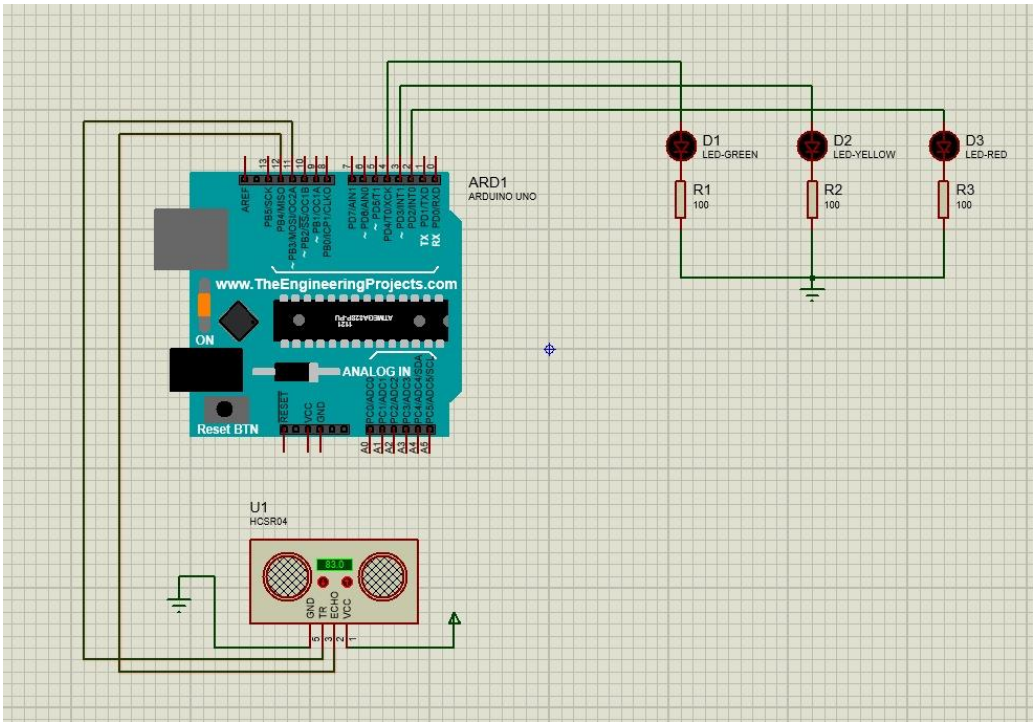

Figure 01: Arduino board's pin connections with the Sonar Sensors and LEDs

Code Explanation:

```
// define the pin numbers
const int trigPin = 11;
const int echoPin = 12;

// define variables
long duration;
float distance, distanceinches, distanceThreshold;

void setup() {
Serial.begin(9600); // Starts the serial communication
pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
pinMode(echoPin, INPUT); // Sets the echoPin as an Input
pinMode(2, OUTPUT); // Sets pins 2, 3, and 4 as the Output pin
pinMode(3, OUTPUT);
pinMode(4, OUTPUT);
}

void loop() {
// Clears the trigPin
digitalWrite(trigPin, LOW);
delayMicroseconds(2);

// Sets the trigPin on HIGH state for 10 microseconds
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

// Reads the echoPin, returns the sound wave travel time in microseconds
duration = pulseIn(echoPin, HIGH);

// Calculating the distance
distance = (duration/2)*1e-6*340*100;
distanceinches = (distance/2.54);

// Prints the distance on the Serial Monitor
Serial.print("Distance = ");
Serial.print(distance);
Serial.print(" cm;  ");
Serial.print("Distance = ");
Serial.print(distanceinches);
Serial.println("inches");

// set threshold distance to activate LEDs
distanceThreshold = 80;
```

```
if (distance > distanceThreshold) {
digitalWrite(2, LOW);
digitalWrite(3, LOW);
digitalWrite(4, LOW);
}

if (distance < distanceThreshold && distance > distanceThreshold-30) {
digitalWrite(2, HIGH);
digitalWrite(3, LOW);
digitalWrite(4, LOW);
}

if (distance < distanceThreshold-30 && distance > distanceThreshold-50) {
digitalWrite(2, HIGH);
digitalWrite(3, HIGH);
digitalWrite(4, LOW);
}

if (distance < distanceThreshold-50 && distance > distanceThreshold-70 ) {
digitalWrite(2, HIGH);
digitalWrite(3, HIGH);
digitalWrite(4, HIGH);
}
delay(200); // Wait for 200 millisecond(s)
}
```

Explanation: This code uses an HC-SR04 ultrasonic sensor to measure the distance of an object and display the proximity level using three LEDs connected to digital pins 2, 3, and 4. The sensor has two main components: a trigger pin (trigPin, pin 11) that sends an ultrasonic pulse, and an echo pin (echoPin, pin 12) that receives the reflected signal from a nearby object. The time between sending and receiving the pulse is used to calculate the distance of the object. In the setup() function, the trigPin is configured as an output and the echoPin as an input. Additionally, digital pins 2, 3, and 4 are set as output pins for the LEDs. Serial communication is started at a baud rate of **9600** to allow real-time monitoring of measured values through the Serial Monitor.In the loop() function, the sensor begins by sending a 10-microsecond HIGH signal to the trigPin. This triggers the sensor to emit an ultrasonic pulse. The echoPin listens for the reflected pulse, and the pulseIn() function measures the time it takes for the echo to return. This duration is converted into distance in centimeters using the formula,with necessary unit conversions. The code also calculates the equivalent distance in inches and prints both values on the Serial Monitor.Based on the calculated distance, the code controls the LEDs to visually represent how far an object is. If the object is **farther than 80 cm** (the defined threshold), **all three LEDs** remain **off**, indicating no nearby object. If the **distance falls between 50 and 80 cm,** the **first LED (pin 2)** turns **ON**. When the object moves closer **(30–50 cm)**, **two LEDs (pins 2 and 3)** are **ON**, and for very close distances **under 30 cm**, **all three LEDs** are turned **ON (pins 2, 3, and 4**). The delay of 200 milliseconds ensures the readings are updated smoothly without flickering.

Hardware Implementation and Explanation:

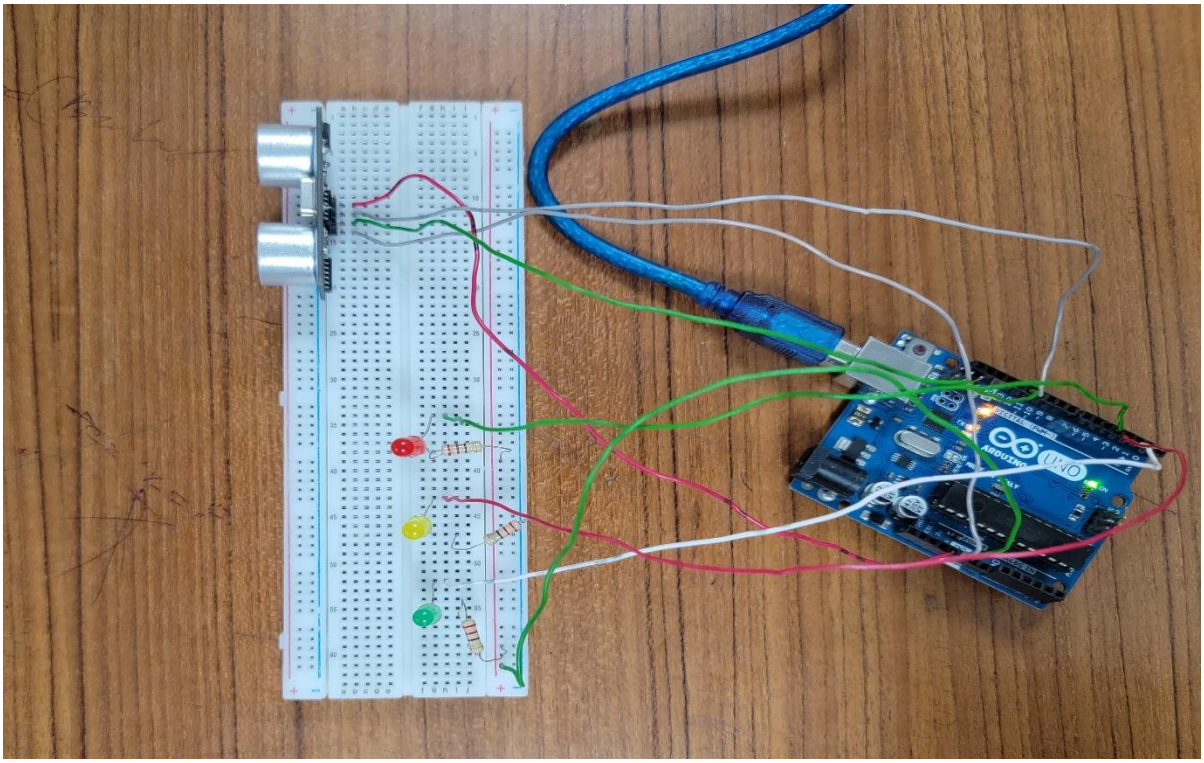

Figure 02: Hardware implementation of an obstacle detection system.

Explanation: In the setup, an Arduino Uno is used as the main controller. The HC-SR04 ultrasonic sensor is connected to the breadboard, with its Trig pin connected to Arduino pin 11 and Echo pin connected to pin 12. These two pins are responsible for sending and receiving ultrasonic pulses. The sensor is powered by connecting its VCC and GND to the 5V and GND pins of the Arduino, respectively.Three LEDs are arranged vertically on the breadboard, each connected to digital pins 2 (green), 3 (yellow), and 4 (red) of the Arduino. Each LED has a current-limiting resistor 100Ω in series to prevent excess current from damaging the LEDs. The longer leg (anode) of each LED connects to the digital output pin through a resistor, and the shorter leg (cathode) connects to the GND rail of the breadboard. The USB cable connected to the Arduino board provides power and allows for code upload and Serial Monitor communication. The wiring setup ensures that based on the object's distance detected by the sensor, the Arduino lights up one or more LEDs to visually represent proximity.

Experimental Output Results:

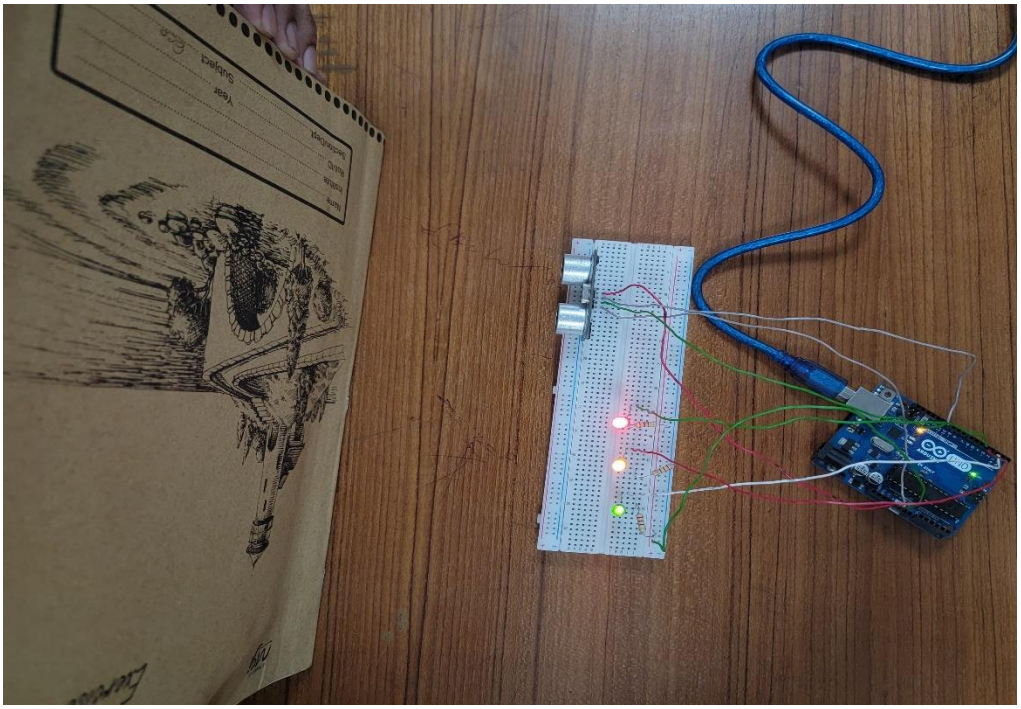

Figure 03: All LEDs on (When the object is under 30cm)

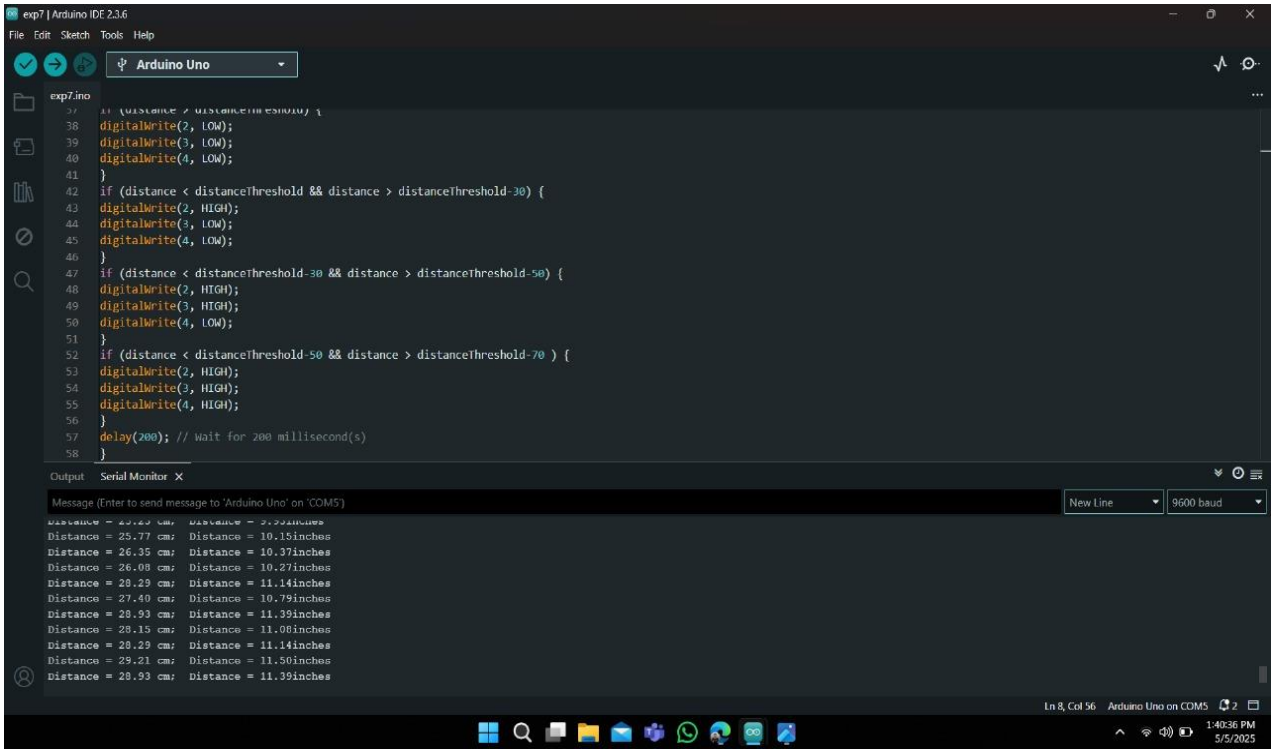

Figure 04: All LEDs on (When the object is under 30cm)

Figure 05: Two LEDs on (When the object moves closer 30-50 cm)

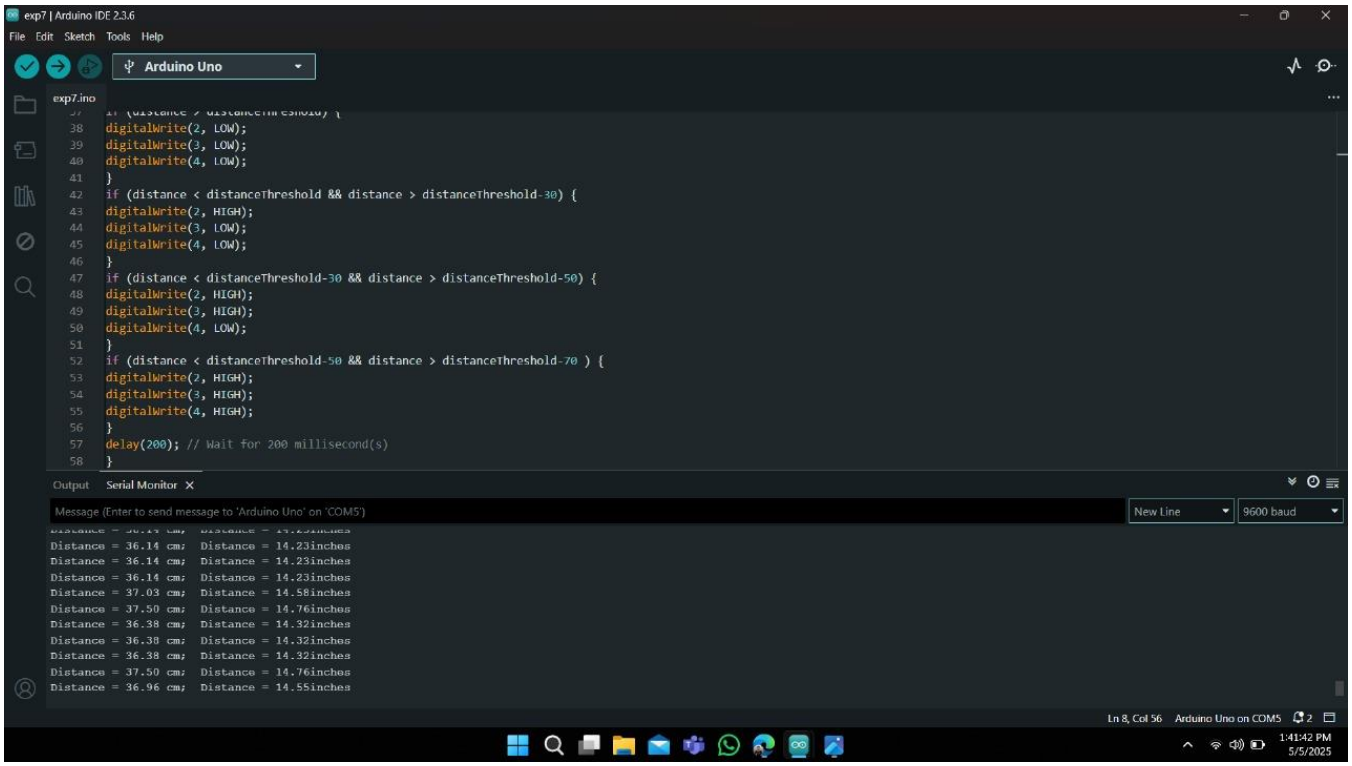

Figure 06: Two LEDs on (When the object moves closer 30-50 cm)

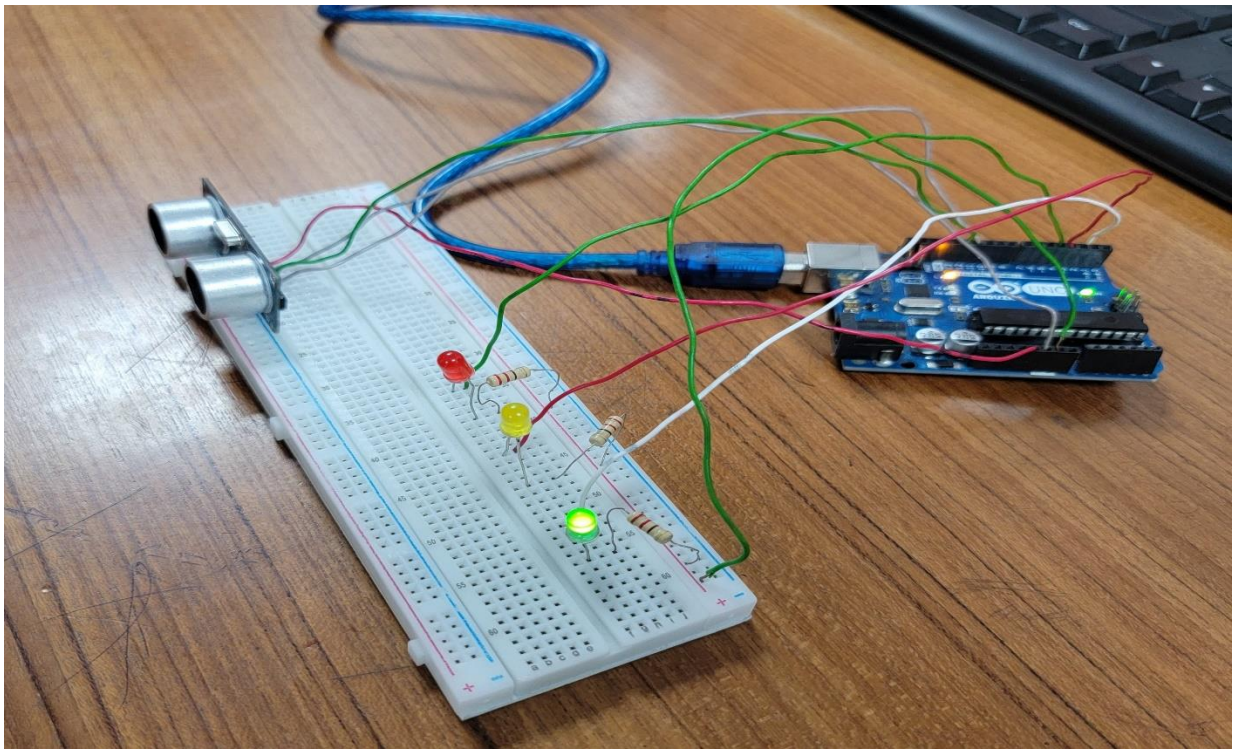

Figure 07: One LED ON (when the object is between 50 and 80 cm away)

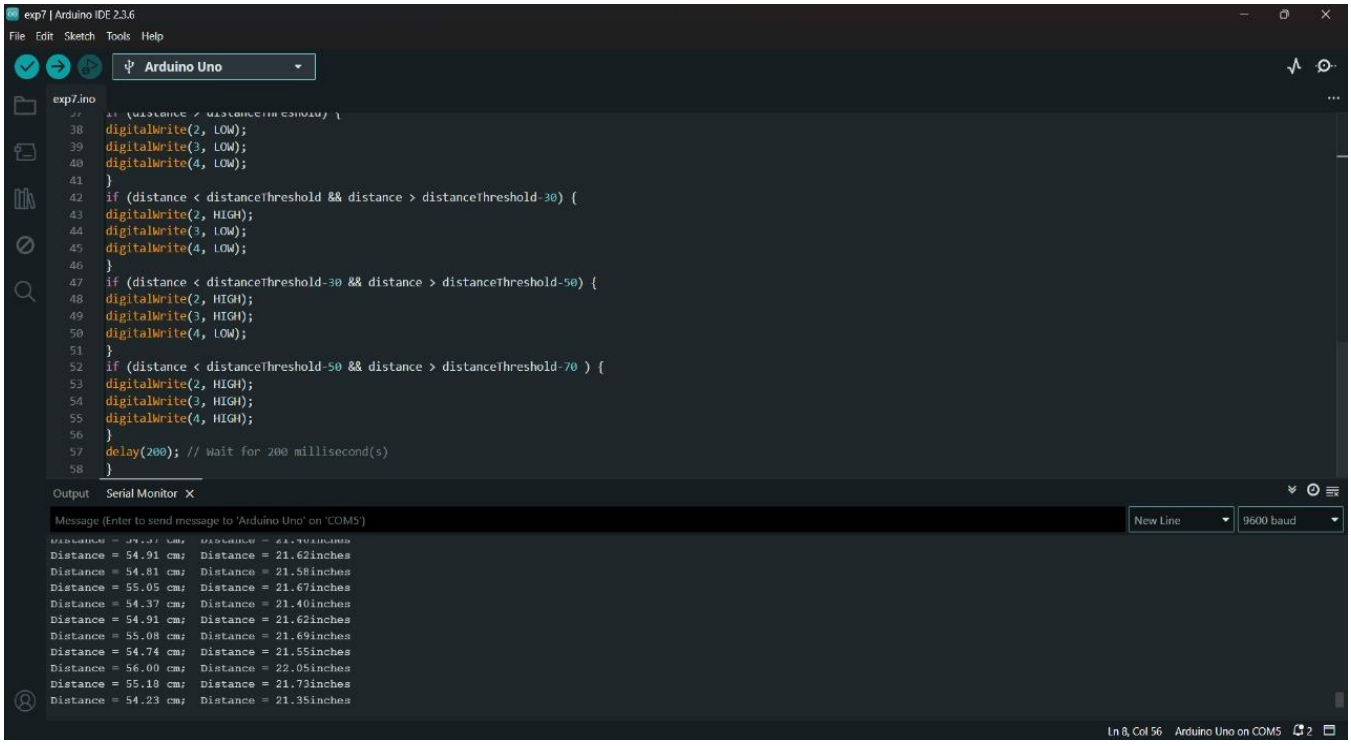

Figure 08: One LED ON (when the object is between 50 and 80 cm away)

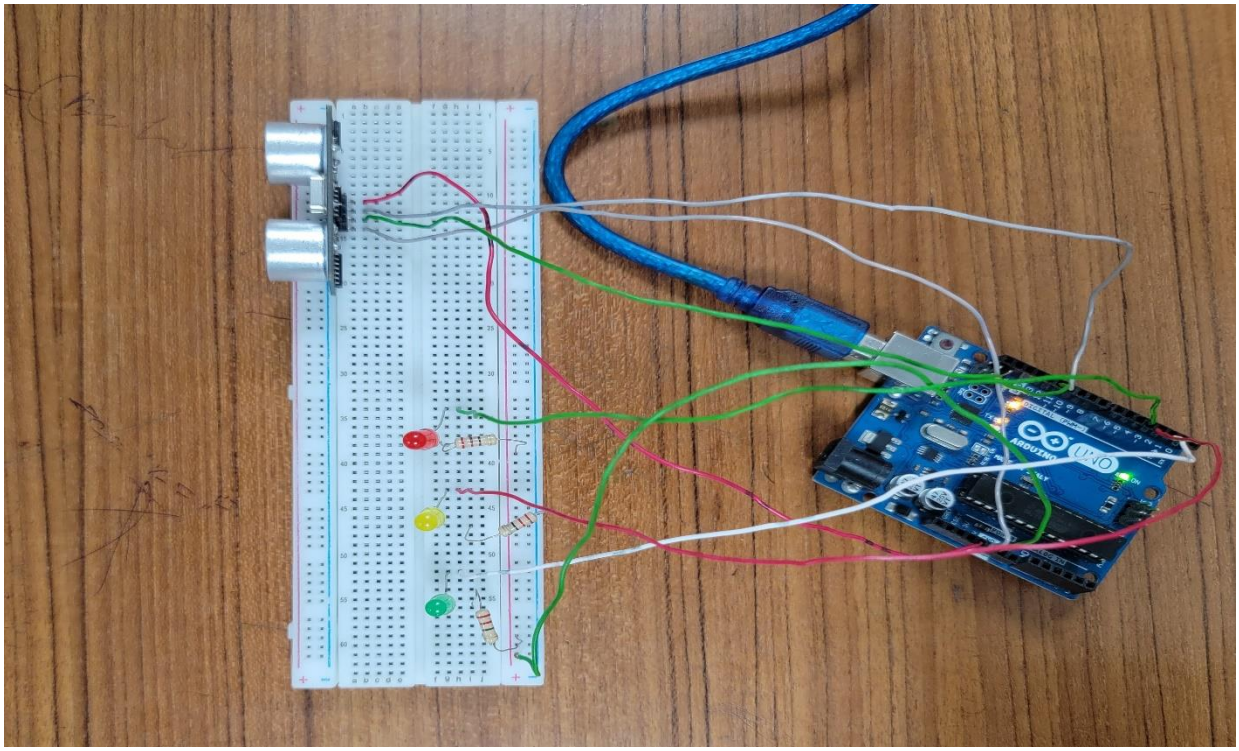

Figure 09: All LEDs OFF (when the object is farther than 80 cm)

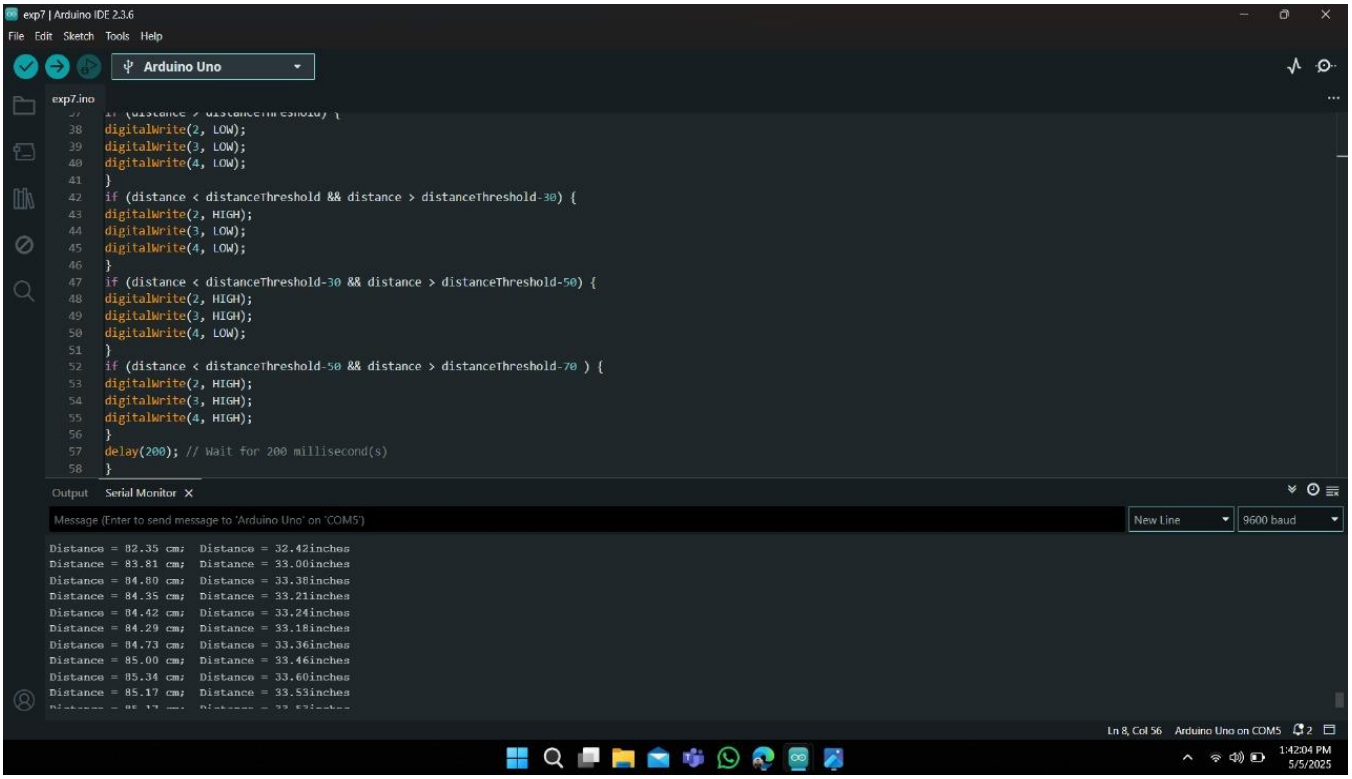

Figure 10: All LEDs OFF (when the object is farther than 80 cm)

Explanation: When the Arduino code is uploaded and the system starts running, the HC-SR04 ultrasonic sensor immediately begins measuring the distance to any object in front of it. The Trig pin (connected to pin 11) sends out an ultrasonic pulse, and the Echo pin (connected to pin 12) listens for the reflected signal. The time taken for the echo to return is captured using the pulseIn() function, and the code calculates the distance using the formula: distance = (duration / 2) × 1e-6 × 340 × 100, converting the time into distance in centimeters. This distance is printed live on the Serial Monitor in both centimeters and inches for monitoring.Based on this real-time distance value, the Arduino controls three LEDs (connected to pins 2, 3, and 4) to indicate proximity levels. If the object comes **closer than 30 cm**, **all three LEDs** (pins 2, 3, and 4) turn **ON**, indicating the object is very near. When the object is **closer, between 30 cm and 50 cm**, **two LEDs** (pins 2 and 3) light **ON**, signaling increasing proximity. If the object moves **between 50 cm and 80 cm**, **only one LED** (pin 2) turns **ON**. When the object is **more than 80 cm** away, the set threshold-**all LEDs** remain **OFF**, showing no object nearby. This dynamic behavior provides a simple yet effective way to understand distance through LED feedback.

Simulation Output Results:

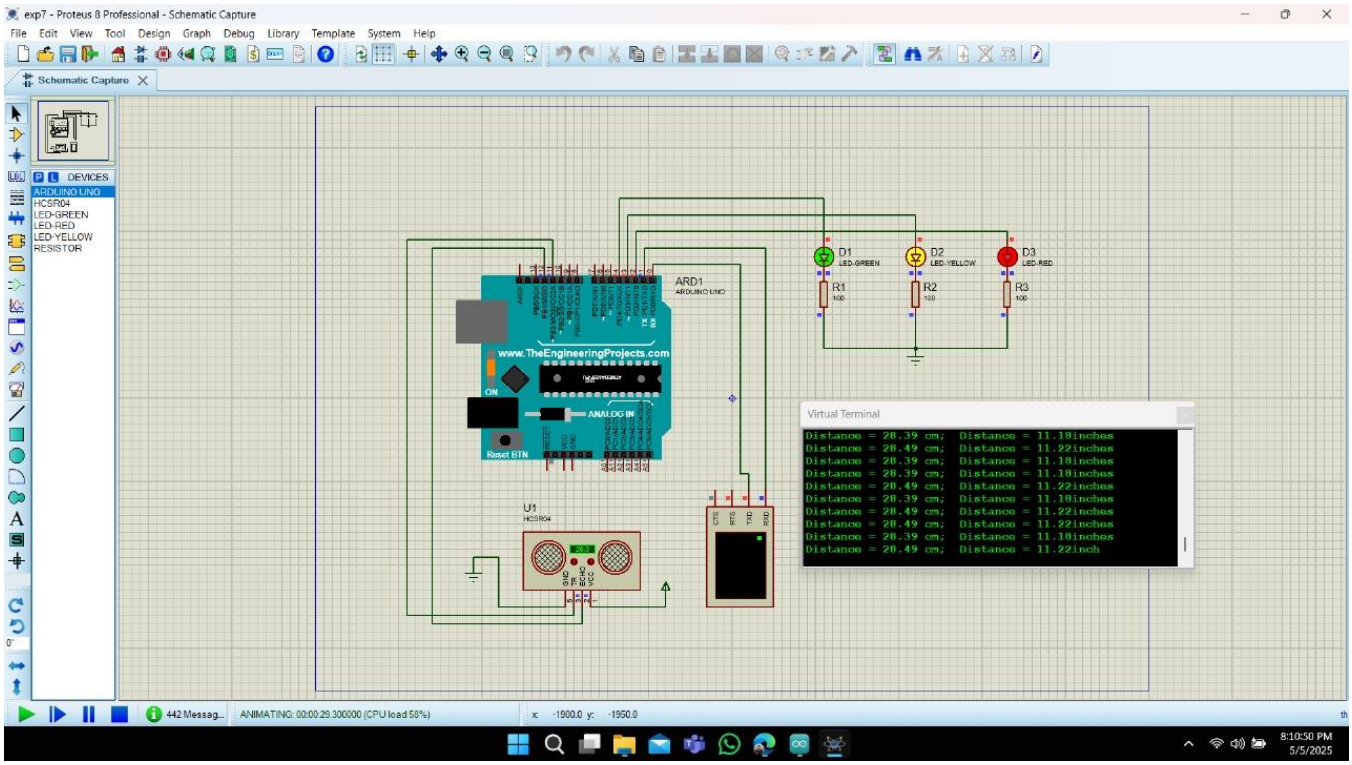

Figure 11: Simulation for all LEDs ON (when the object is closer than 30 cm)

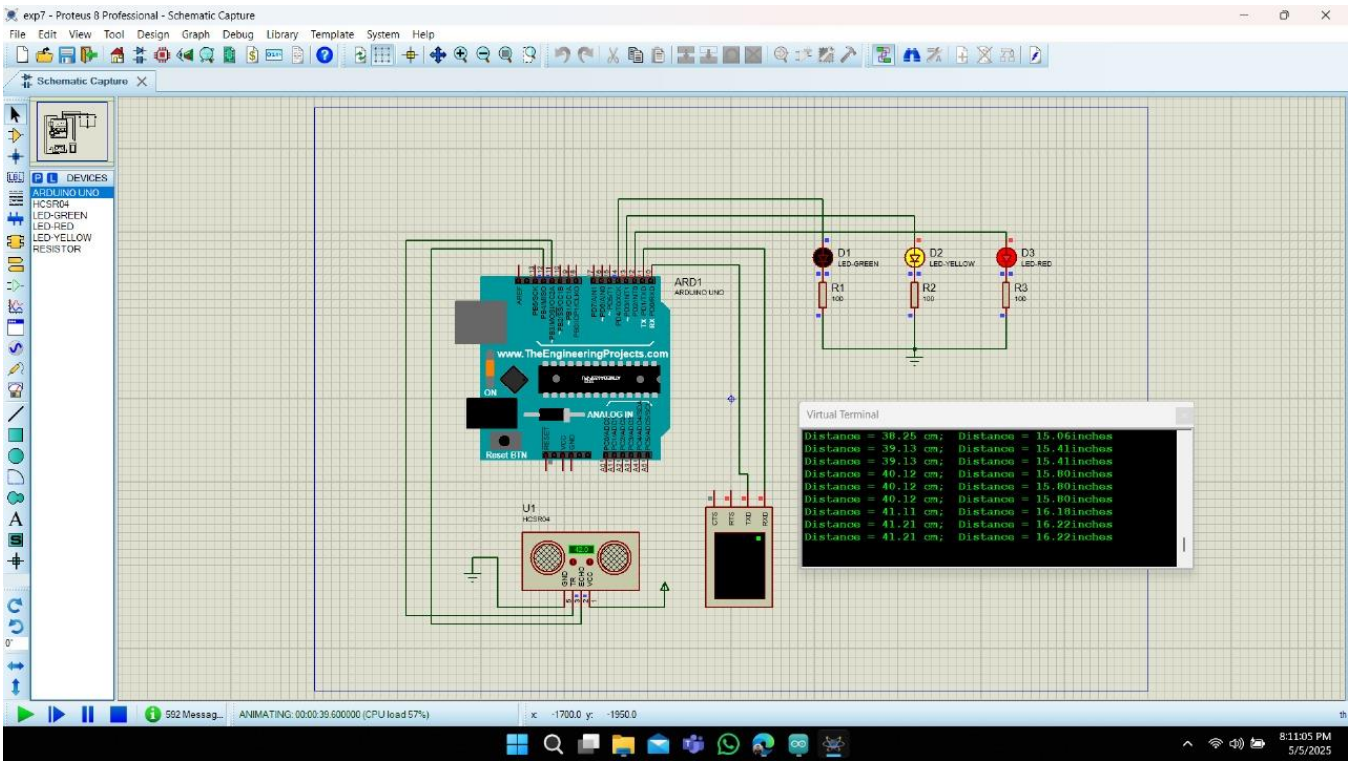

Figure 12: Simulation for two LEDs ON (when the object is within 30–50 cm)

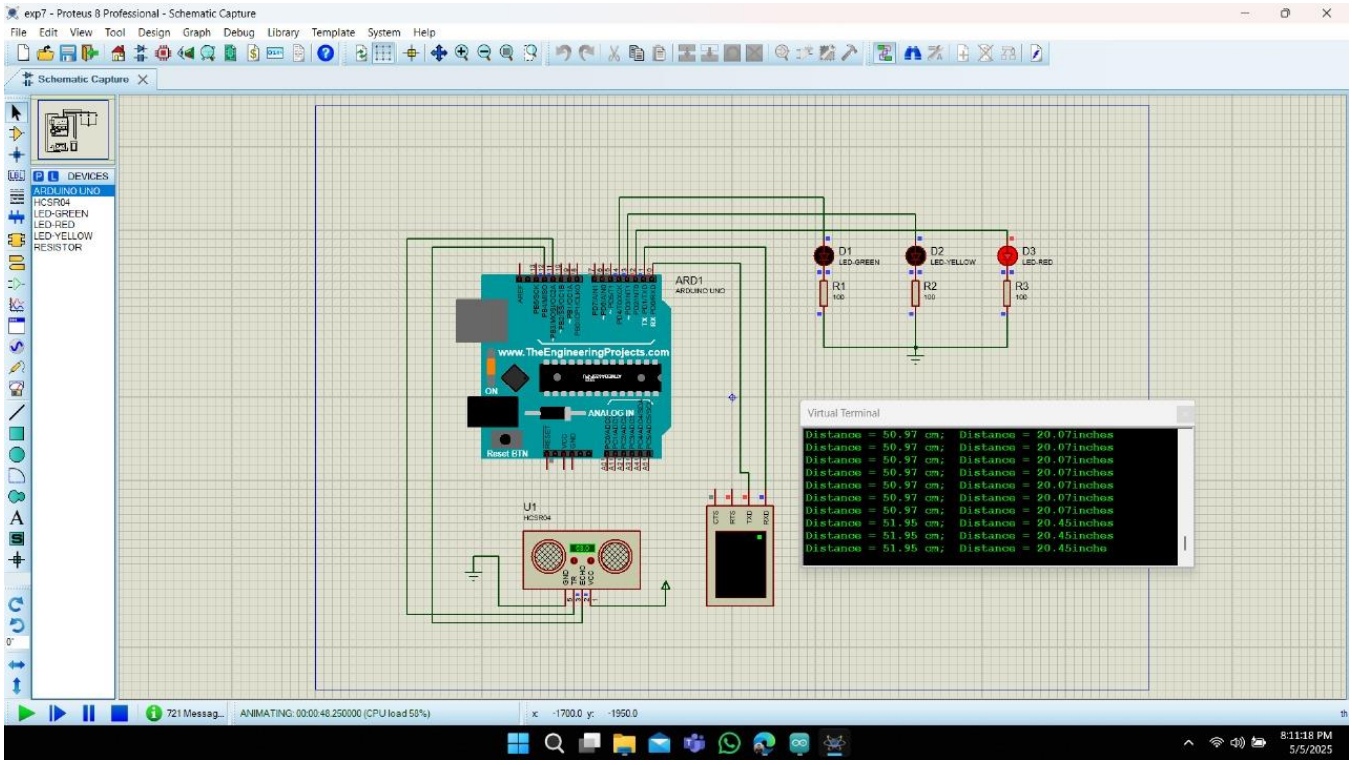

Figure 13:Simulation for one LED ON (when the object moves closer between 50–80 cm)

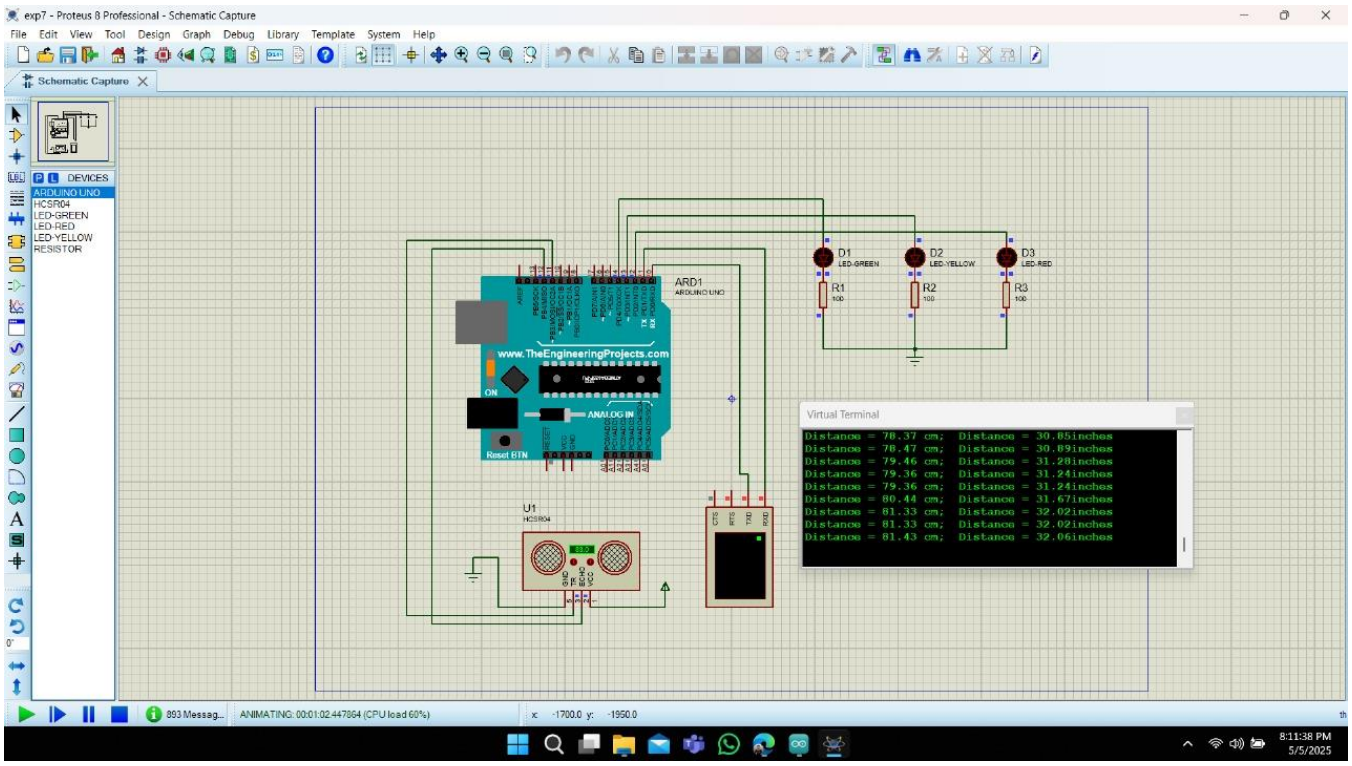

Figure 14: Simulation for all LEDs OFF (when the object is farther than 80 cm)

Explanation: In this simulation, the Arduino Uno board, three 100Ω resistors(Green, Yellow,Red)LEDs and a sonar sensor were set up according to the hardware configuration used in the lab. The program was written and verified in the Arduino IDE 2.3.5, which generated a HEX file. This file was then loaded into the Proteus simulation to simulate the behavior of the circuit. After running the simulation, the results were observed and compared with the actual hardware results to check for consistency and verify the performance of the system.

Answer to Question:

3) My ID no- 23-51269-1, Based on my ID , the output port should be Pin 1. However, Pin 1 is the **TX pin used for sending data via Serial Monitor**, so using it for output caused a conflict. To fix this and ensure proper serial output, I used **Pin 4** for the output instead.

According to question P=5(echoPin as an Input), Q= ~~1~~/4(trigPin as an Output) and A=2,B=6,C=9(Output Pins for LEDs).

Code:

```
// define the pin numbers
const int trigPin = 4;
const int echoPin = 5;

// define variables
long duration;
float distance, distanceinches, distanceThreshold;
```

```
void setup() {
Serial.begin(9600); // Starts the serial communication
pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
pinMode(echoPin, INPUT); // Sets the echoPin as an Input
pinMode(2, OUTPUT); // Sets pins 2, 6, and 9 as the Output pin
pinMode(6, OUTPUT);
pinMode(9, OUTPUT);
}
void loop() {
// Clears the trigPin
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
// Sets the trigPin on HIGH state for 10 microseconds
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
// Reads the echoPin, returns the sound wave travel time in microseconds
duration = pulseIn(echoPin, HIGH);
// Calculating the distance
distance = (duration/2)*1e-6*340*100;
distanceinches = (distance/2.54);
// Prints the distance on the Serial Monitor
Serial.print("Distance = ");
Serial.print(distance);
Serial.print(" cm;  ");
Serial.print("Distance = ");
Serial.print(distanceinches);
Serial.println(" inches");
// set threshold distance to activate LEDs
distanceThreshold = 80;
if (distance > distanceThreshold) {
digitalWrite(2, LOW);
digitalWrite(6, LOW);
digitalWrite(9, LOW);
}
if (distance < distanceThreshold && distance > distanceThreshold-30) {
digitalWrite(2, HIGH);
digitalWrite(6, LOW);
digitalWrite(9, LOW);
}
if (distance < distanceThreshold-30 && distance > distanceThreshold-50) {
digitalWrite(2, HIGH);
digitalWrite(6, HIGH);
digitalWrite(9, LOW);
}
if (distance < distanceThreshold-50 && distance > distanceThreshold-70 ) {
digitalWrite(2, HIGH);
digitalWrite(6, HIGH);
digitalWrite(9, HIGH);
}
delay(200); // Wait for 200 millisecond(s) }
```

Explanation: This code uses an HC-SR04 ultrasonic sensor to measure the distance of an object and display the proximity level using three LEDs connected to digital pins 2, 6, and 9. The sensor has two main components: a trigger pin (trigPin, pin 4) that sends an ultrasonic pulse, and an echo pin (echoPin, pin 5) that receives the reflected signal from a nearby object. The time between sending and receiving the pulse is used to calculate the distance of the object. In the setup() function, the trigPin is configured as an output and the echoPin as an input. Additionally, digital pins 2, 6, and 9 are set as output pins for the LEDs. Serial communication is started at a baud rate of **9600** to allow real-time monitoring of measured values through the Serial Monitor.In the loop() function, the sensor begins by sending a 10-microsecond HIGH signal to the trigPin. This triggers the sensor to emit an ultrasonic pulse. The echoPin listens for the reflected pulse, and the pulseIn() function measures the time it takes for the echo to return. This duration is converted into distance in centimeters using the formula,with necessary unit conversions. The code also calculates the equivalent distance in inches and prints both values on the Serial Monitor.Based on the calculated distance, the code controls the LEDs to visually represent how far an object is. If the object is **farther than 80 cm** (the defined threshold**), all three LEDs** remain **OFF**, indicating no nearby object. If the **distance falls between 50 and 80 cm**, the **first LED (pin 2)** turns **ON**. When the object moves closer **(30–50 cm)**, **two LEDs (pins 2 and 6)** are **ON**, and for very close distances **under 30 cm**, **all three LEDs** are turned **ON (pins 2, 6, and 9).** The delay of 200 milliseconds ensures the readings are updated smoothly without flickering.

Simulation Results:

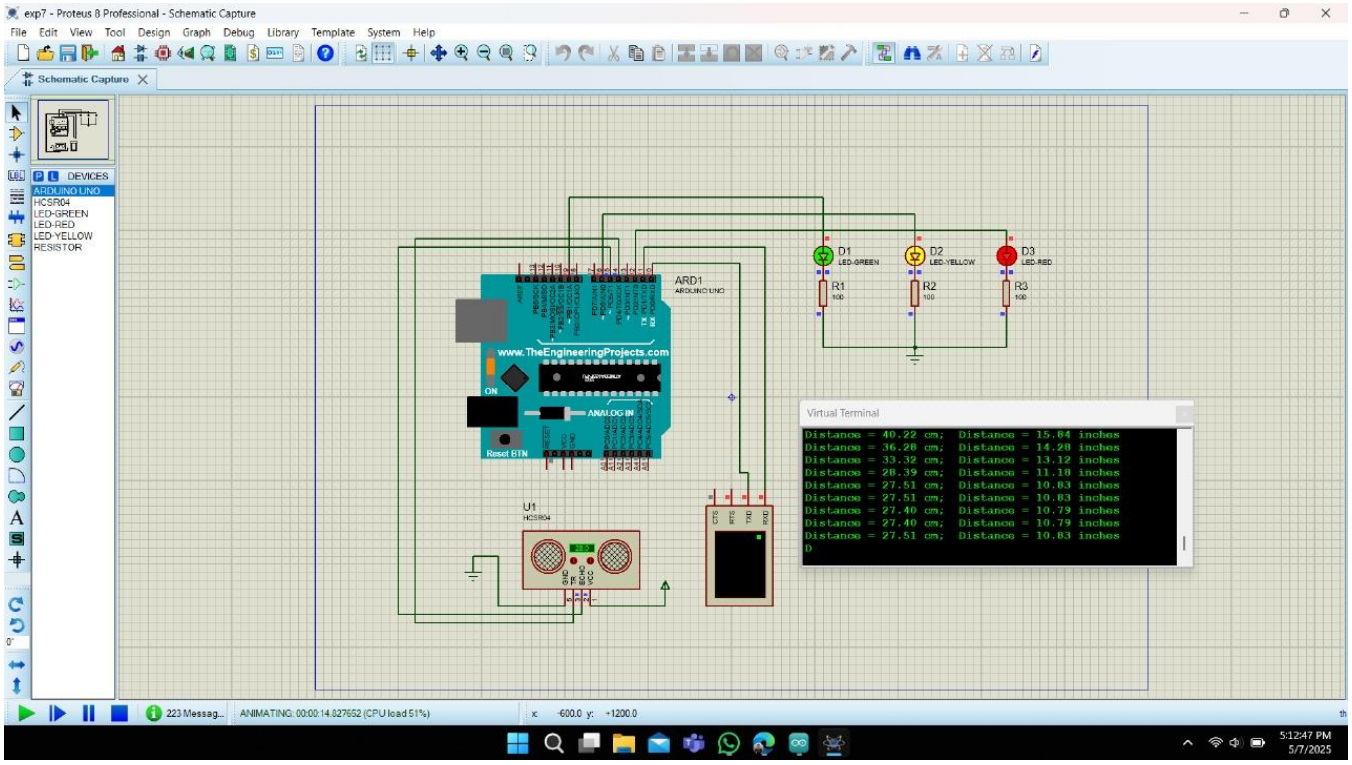

Figure 15: Simulation for all LEDs ON (when the object is closer than 30 cm)

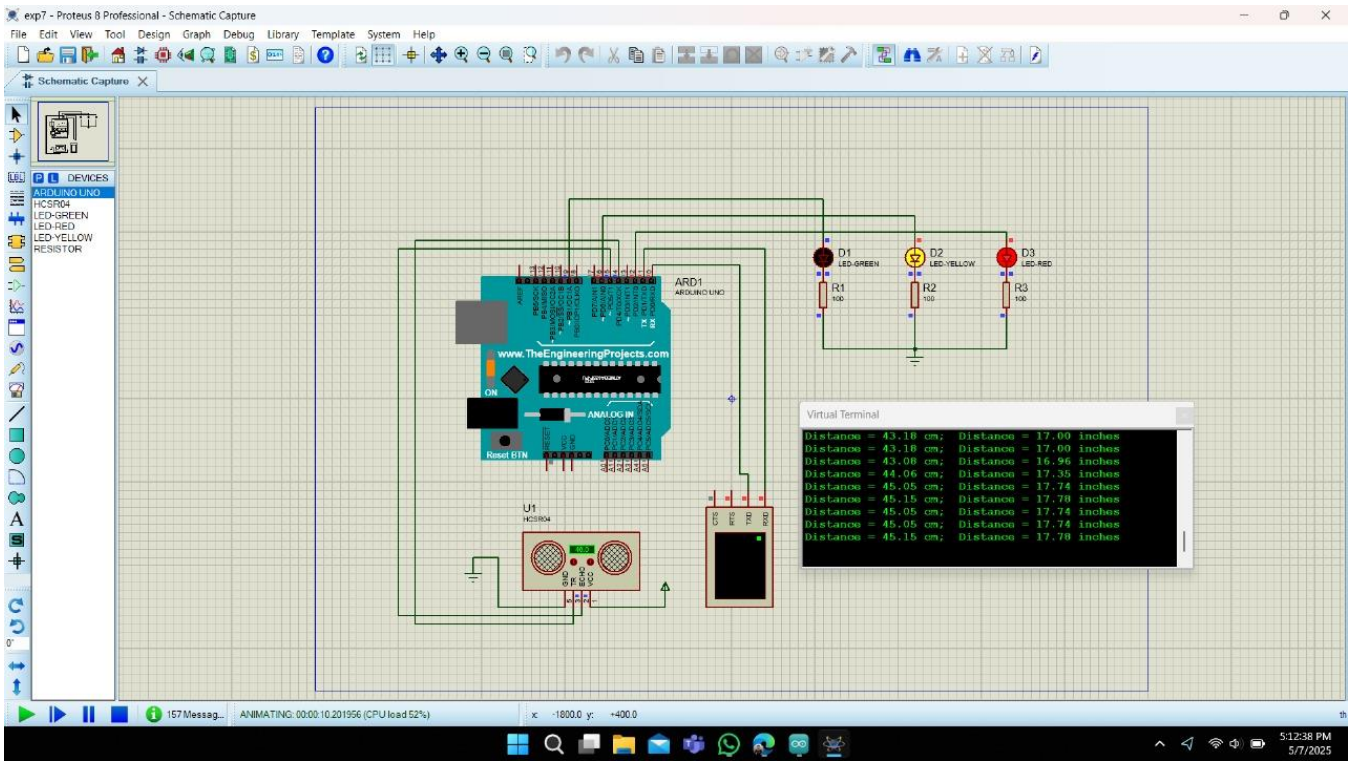

Figure 16: Simulation for two LEDs ON (when the object is within 30–50 cm)

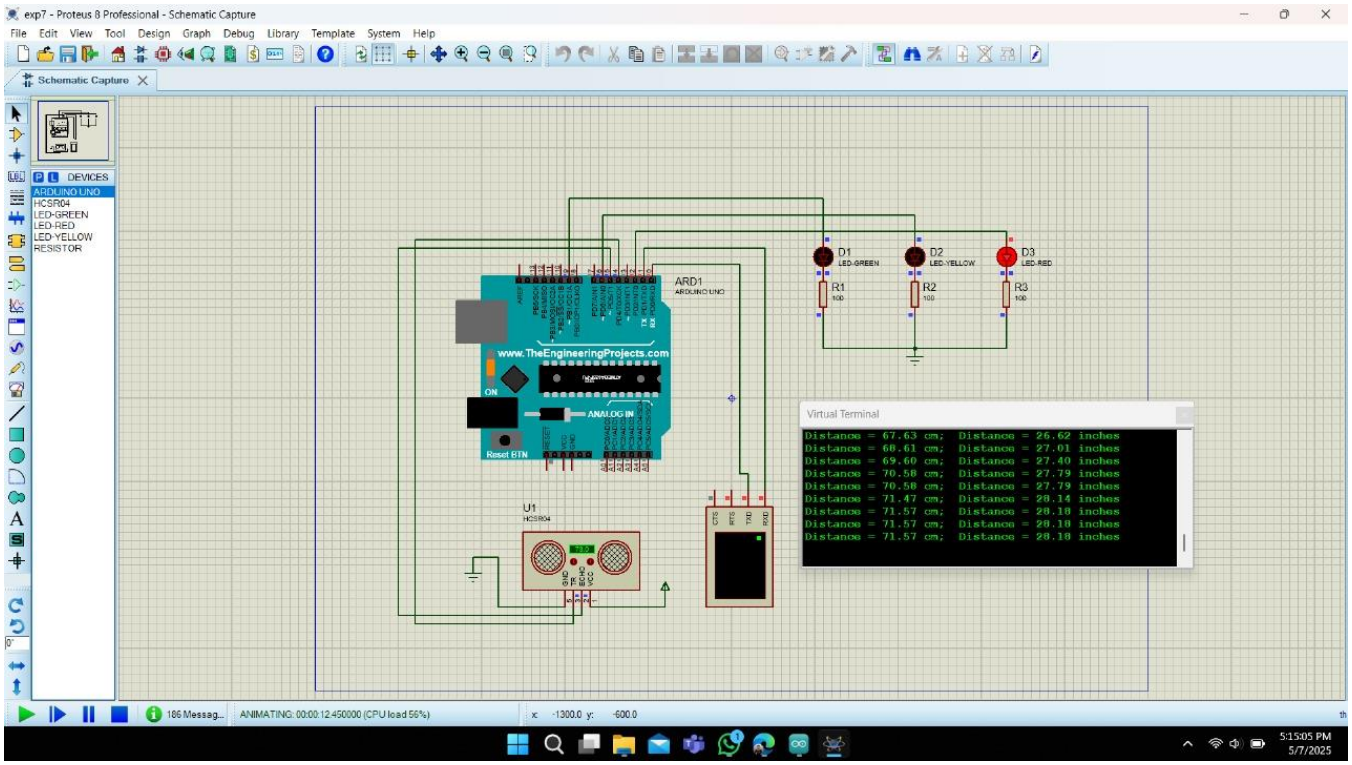

Figure 17: Simulation for one LED ON (when the object moves closer between 50–80 cm)

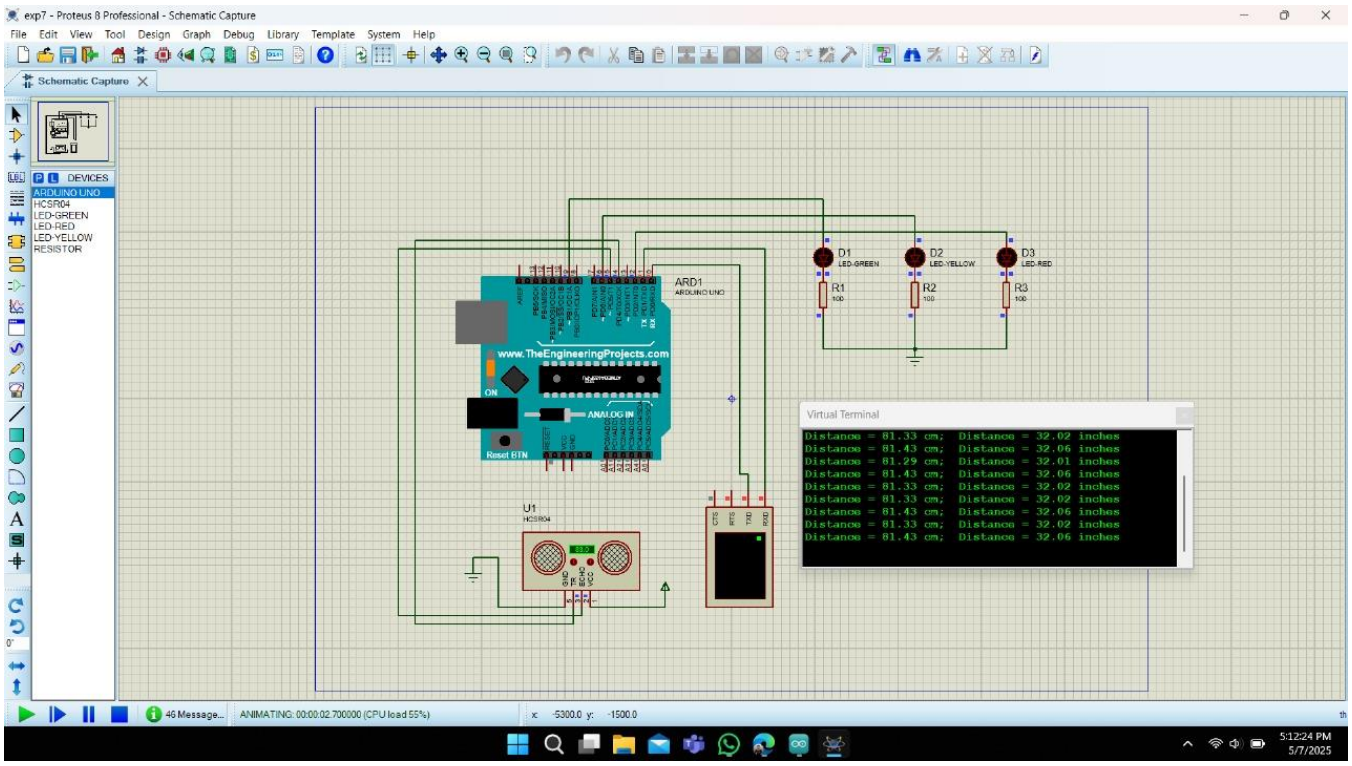

Figure 18: Simulation for all LEDs OFF (when the object is farther than 80 cm)

Explanation: In this simulation, the Arduino Uno board, three 100Ω resistors(Green, Yellow,Red)LEDs were set up according to the hardware configuration used in the lab. The program was written and verified in the Arduino IDE 2.3.5, which generated a HEX file. This file was then loaded into the Proteus simulation to simulate the behavior of the circuit. After running the simulation, the results were observed and verify the performance of the system.

Discussion:

This experiment focused on gaining familiarity with serial communication protocols and external sensor interfacing with an Arduino Microcontroller Board by implementing an obstacle detection system using an ultrasonic sensor. The objective was to explore how distance measurements can be accurately obtained and interpreted in real time, and how serial communication can be used to monitor and debug sensor performance.

In the setup, the ultrasonic sensor was connected to the Arduino through digital pins for triggering and echo reception. The Arduino generated ultrasonic pulses and measured the time it took for the echo to return, then used that time to calculate the distance to the nearest object. Unlike fixed delay-based systems, this approach dynamically responded to environmental changes. The distance was printed to the Serial Monitor in both centimeters and inches using Serial.print(), providing a continuous live data feed for debugging and system evaluation. This made the system interactive and useful for monitoring through a computer interface. A threshold-based system using three LEDs was implemented to visually indicate the proximity of obstacles. The LEDs lit up sequentially as the object approached closer to the sensor, allowing users to quickly assess the situation. This provided a practical understanding of real-time decision-making in embedded systems based on sensor data.

This experiment provided valuable hands-on experience in interfacing an ultrasonic sensor with an Arduino microcontroller using serial communication, which is fundamental in building interactive and intelligent embedded systems. By capturing real-time distance data through the sensor and processing it with Arduino logic, we learned how to implement obstacle detection and visual feedback using LEDs based on proximity levels. The use of serial communication allowed for clear monitoring of sensor readings, aiding in debugging and system calibration. Through this setup, we enhanced our understanding of sensor behavior, analog-to-digital data handling, and real-time decision-making in microcontroller applications. Furthermore, this experiment reinforced key concepts such as conditional statements, threshold-based responses, hardware pin configuration, and non-blocking feedback mechanisms. These skills are not only essential in academic projects but also widely applied in real-world systems like robotics, automation, smart vehicles, and assistive technologies.

Here are some real life scenario and application of this experiment:

1.Automated Parking Assistance Systems: Microcontroller-based obstacle detection helps vehicles detect nearby objects when parking.

Example*:* When reversing, the ultrasonic sensor detects distance from walls or other cars and activates LEDs or buzzers based on how close the obstacle is, reducing the chance of collision.

2. Robot Navigation and Obstacle Avoidance: Autonomous robots use ultrasonic sensors to detect and avoid obstacles in real time.

Example: In line-following or room-cleaning robots, Arduino reads sensor data and adjusts the robot's path to prevent crashing into walls or objects.

3. Blind Spot Detection for Safety: Ultrasonic sensors installed in vehicles or machinery detect objects in the driver/operator's blind spots.

Example: When someone or something comes too close to the side or rear of the vehicle, the system alerts the driver using lights or sound.

4. Security and Intrusion Detection Systems: Arduino-based sensors monitor restricted areas and detect motion or the presence of an intruder.

Example: When a person walks into a protected zone, the system triggers a light or sound alarm and logs the event through serial communication.

5. Smart Trash Bins: Distance sensors detect the fill level of trash bins to signal when emptying is required.

Example: When trash reaches a certain level, the Arduino sends a signal to an indicator or remotely logs data through serial communication for collection teams.

6. Automated Door Systems: Doors that open and close based on proximity detection use ultrasonic sensors for control.

Example: As a person approaches a store entrance, the Arduino reads the sensor and opens the door if the detected distance is below a threshold.

7. Collision Avoidance in Drones: Ultrasonic sensors help drones avoid mid-air obstacles during navigation.

Example: During flight, if the drone detects an obstacle ahead, the system adjusts the flight path to avoid a crash, using real-time sensor data.

8. Assistance Devices for the Visually Impaired: Wearable obstacle detection systems alert users of nearby objects.

Example: An Arduino-powered device vibrates or beeps if it detects an object too close, helping visually impaired individuals walk more safely.

Conclusion:

In conclusion, this experiment provided valuable insight into the fundamentals of embedded systems and sensor-based automation. By integrating an ultrasonic sensor with an Arduino board, we were able to measure real-time distances and control LED indicators based on proximity thresholds. The project highlighted the effective use of serial communication for monitoring data, and showcased how conditional logic can drive responsive behavior in hardware. Through this hands-on implementation, we learned how to convert time-based sensor data into meaningful distance values and apply decision-making logic to control outputs. The experiment reinforced key concepts such as interfacing sensors, real-time data processing, and threshold-based actions. It also demonstrated how simple components can be used to create an efficient obstacle detection system. This experience lays the groundwork for future development of intelligent systems in fields such as robotics, automotive safety, and IoT-based automation, where real-time monitoring and control are essential for smart and adaptive solutions.

References:

[1] **https://www.arduino.cc/.**
[2] **https://www.educba.com**
[3] **https://www.researchgate.net/publication/**
[4] **https://www.geeksforgeeks.org**