Documentation For Distance Measurement System

Project Title >> Distance Measurement with Arduino Uno Using HC-SR04, I2C LCD Display and LED Indicator

Name	Date
Min Thant	7 May . 2025

Table of Contents

1. Introduction

This project implements an embedded distance detection system using the HC-SR04 ultrasonic sensor, an LED indicator, and a 16x2 I2C LCD for real-time distance feedback. The system continuously monitors the proximity of an object and activates a visual LED indicator based on a defined threshold (30 cm). This forms the basis of many smart IoT systems such as:

- Obstacle detectors in robotics
- Automated parking assistants
- Social distancing monitors
- Security/alarm systems

---

2. Objectives

This project aims to create a real-time distance monitoring system using an HC-SR04 ultrasonic sensor, Arduino, I2C LCD, and an LED indicator. The specific objectives are:

Measure Distance Accurately
Use the ultrasonic sensor to detect the distance of nearby objects and calculate the value using time-of-flight principles based on the speed of sound.

Thredshold-Based Decision Making
Compare the measured distance to a preset threshold (30 cm). If the object is closer than or equal to this value, the system will trigger a visual alert.

Visual Feedback with LED
Control an LED to indicate system status:

- **ON**: Object is farther than 30 cm (safe).
- **OFF**: Object is too close (alert).

Display Data on I2C LCD
Show the measured distance and system status clearly on a 16x2 LCD for user-friendly feedback.

Serial Output for Monitoring
Send distance readings and system messages to the Serial Monitor for debugging and analysis.

Efficient and Expandable Code
Maintain clean, modular code using separate functions and defined variables, making the system easy to expand or modify.

---

3. Hardware & Software

3.1. Hardware Components

Component	Specifications	Technical Role in System
Arduino Uno	ATmega328P, 16 MHz, 14 digital I/O, 6 analog inputs	Core microcontroller platform used to execute the program logic, read sensor data, and control outputs.
HC-SR04 Ultrasonic Sensor	Range: 2–400 cm, Accuracy: ±3 mm, 40 kHz ultrasound	Measures distance by sending ultrasonic pulses and measuring time until echo is received.
16x2 LCD with I2C Backpack	Operating voltage: 5V, I2C address: 0x27, SDA/SCL interface	Displays real-time distance and LED status; I2C interface reduces pin usage (only 2 pins needed).
LED (5mm)	Forward voltage: ~2V, Forward current: 20 mA	Provides visual indication based on the distance threshold condition (ON/OFF status).
Resistor (220Ω)	Tolerance: ±5%, Power Rating: 0.25W	Limits current through the LED to prevent damage (Ohm's Law: V=IR).
Breadboard	Standard 830-tie point (optional)	Used for temporary circuit assembly and prototyping.
Jumper Wires	Male-to-male, 20–22 AWG	Facilitates electrical connections between Arduino, sensors, and display.
Power Supply	5V via USB or external battery (e.g., 9V with regulator)	Provides required voltage to Arduino and peripherals.

3.2. Software Components

Component	Description
Arduino IDE	Open-source development environment used to write, compile, and upload C/C++ based code to the Arduino microcontroller.
Arduino Core Libraries	Includes digitalWrite(), pinMode(), Serial.begin(), pulseIn(), etc., which provide direct access to microcontroller I/O and peripherals.
LiquidCrystal_I2C.h	A high-level library that simplifies interfacing with an I2C-connected HD44780 LCD module. It abstracts low-level I2C communication using the Wire library.
Serial Monitor (UART)	Provides a terminal interface via USB using UART at 9600 baud rate. It outputs real-time data for monitoring and debugging.

4. Working of HCSR-04 Ultrasonic Sensor

4.1. Main Works

- Emission
- Transmission
- Reflection
- Reception
- Time Measurement
- Distance Calculation

4.2. Emission

The process begins when the microcontroller sends a HIGH signal to the TRIG pin of the sensor for at least 10 microseconds. This activates the piezoelectric transducer inside the sensor, which emits an ultrasonic sound wave at a frequency of 40 kHz well above the human hearing range.

4.3. Transmission

Once emitted, the ultrasonic wave propagates through the air at the speed of sound (~343 meters per second in air at 20°C). The wave continues to travel until it hits a solid object or surface in its path.

4.4. Reflection

When the wave reaches an obstacle, it reflects back toward the sensor. The amount of reflected sound and the quality of the echo depend on the material, shape, and angle of the object.

4.5. Reception

The reflected sound wave (echo) is detected by a second piezoelectric transducer in the sensor, which acts as the receiver. This component converts the returning ultrasonic signal into an electrical pulse.

4.6. Time Measurement

The sensor then calculates the duration between the moment the sound was emitted and the moment the echo was received. This duration is measured in microseconds (µs) using Arduino's pulseIn() function, which captures the time for which the ECHO pin stays HIGH.

4.7. Distance Measurement

The distance between sensor and obstacle can be used using the time gap in previous step. The measuring formula is:

Distance = (Time taken by the sound wave * Speed of sound) / 2.

Speed of sound equal to $343ms^{-1}$ at normal room temperature. To measure the distance in cm, it should convert to cm/ µs.

Speed of sound = 343 m/s

= 343 * (100 cm / 10^6 µs)

= 0.034 cm / µs

Assume that time taken by the sound wave is 1500 µs,

Distance = (1500 * 0.034)/2

= 25.2 cm

---

5. System Design Diagrams

The system is designed to continuously monitor the distance of an object using an ultrasonic sensor and provide both visual alerts (via LED and LCD) and serial feedback. It follows a modular embedded design approach with clearly defined functional blocks for sensing, processing, and output.

5.1. Functional Blocks

Input Block

- HC-SR04 Ultrasonic Sensor acts as the input device.
- It emits a 40 kHz ultrasonic wave and waits for the reflected echo.
- Time between trigger and echo is used to calculate the distance to the object.

Processing Block

- Arduino Microcontroller (Uno) is the control unit.
- It triggers the ultrasonic sensor, reads echo duration, calculates distance, and applies logical decisions based on threshold (30 cm).
- It formats data for both the LCD and Serial Monitor.

Output Block

- LED Indicator: Controlled by digital output pin. Turns ON or OFF depending on distance.
- I2C LCD Display (16x2): Displays real-time distance and LED status.
- Serial Monitor: Outputs diagnostic messages and distance readings for debugging or logging.

5.2. Pin Connections

Ultrasonic Sensor (HC-SR04)

- VCC → 5V
- GND → GND
- TRIG → Pin 13
- ECHO → Pin 12

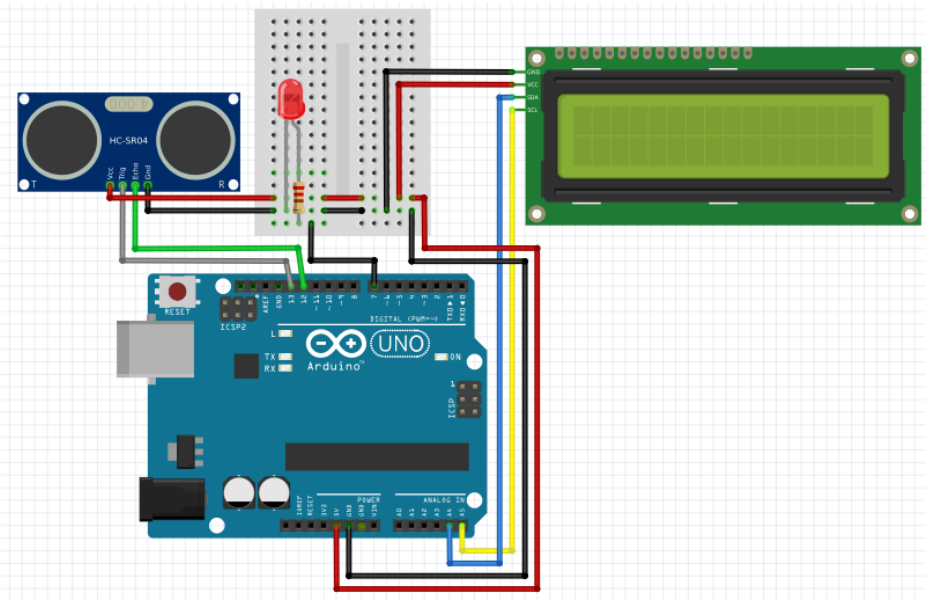
LED

- Anode (+) → Pin 7 (with a series of 220Ω resistor)
- Cathode (–) → GND

I2C LCD (0x27)

- SDA → A4 (Arduino Uno)
- SCL → A5 (Arduino Uno)
- VCC → 5V
- GND → GND

5.3. System Wiring Diagram



---

6. Source Code

```
#include <LiquidCrystal_I2C.h>  // Include the I2C LCD library

// Define pins for ultrasonic sensor
int trig = 13;
int echo = 12;

// Variables to store duration and distance
long duration;
float distance;
```

```
// Define LED pin
int led = 7;

// Initialize the LCD at I2C address 0x27 with 16 columns and 2 rows
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup () {
  pinMode(trig, OUTPUT);            // Set trigger pin as output
  pinMode(echo, INPUT);             // Set echo pin as input
  pinMode(led, OUTPUT);             // Set LED pin as output
  Serial.begin(9600);               // Start serial communication at 9600 baud
  lcd.init();                       // Initialize the LCD
  lcd.backlight();                  // Turn on the LCD backlight
}

// Function to measure distance using ultrasonic sensor
void ultrasonic () {
  digitalWrite(trig, HIGH);           // Send a 10 microsecond pulse
  delayMicroseconds(10);
  digitalWrite(trig, LOW);
  duration = pulseIn(echo, HIGH);   // Measure how long the echo takes to return
  distance = duration * 0.034 / 2;      // Convert time into distance in cm
}

void loop() {
  ultrasonic();  // Call function to update distance

  // If object is closer than or equal to 30 cm
  if (distance <= 30) {
    digitalWrite(led, LOW);  // Turn OFF LED

    // Display distacne on LCD
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Distance=");
    lcd.print(distance);
    lcd.print("cm");
    lcd.setCursor(3, 1);
    lcd.print("LED is OFF!");

    // Print distance to serial
    Serial.print("Distance is ");
    Serial.print(distance);
    Serial.print("cm.");
    Serial.println("LED is OFF!");
    delay(1000);  // Wait for 1 second
  }

  // If object is farther than 30 cm
  else {
    digitalWrite(led, HIGH);  // Turn ON LED

    // Display distacne on LCD
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Distance is over");
    lcd.setCursor(3, 1);
```

```
    lcd.print("LED is ON!");

    // Print distance to serial
    Serial.print("Distance is out of range. Distance is ");
    Serial.print(distance);
    Serial.print("cm.");
    Serial.println("LED is ON!");
    delay(1000);  // Wait for 1 second
  }
}
```

---

7. Code Explanation

7.1. Libraries

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

- This line includes the LiquidCrystal_I2C library, which provides functions for controlling an LCD screen via the I2C protocol.
- Why I2C? It uses only two wires (SDA, SCL) instead of many pins, saving space on the Arduino board.

7.2. Global Declarations

```
int trig = 13;
int echo = 12;
```

- These two variables define which Arduino digital pins are connected to the HC-SR04 ultrasonic sensor.
    - Trig pin (output): Arduino sends a high pulse to start the distance measurement.
    - Echo pin (input): Reads how long it takes the sound wave to return.

```
long duration;
float distance;
```

- duration : stores the time in microseconds that the echo pulse lasted.
- distance: stores the calculated distance in centimeters using the speed of sound formula.

```
int led = 7;
```

- Declares the digital pin connected to an LED. This LED is used to signal whether the object is close (LED OFF) or far (LED ON).

7.3. LCD Object Initialization

```
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

- Initializes an LCD object named lcd.

    - 0x27: The I2C address (often 0x27 or 0x3F for common displays).
    - 16, 2: The display has 16 columns and 2 rows.
- This object provides functions to print text and control the screen.

7.4. Setup Function

```
void setup () {
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
  pinMode(led, OUTPUT);
```

- Sets up hardware pin modes:
    - trig: Arduino will send a signal → OUTPUT
    - echo: Arduino will receive a signal → INPUT
    - led: Will be turned on/off based on object distance → OUTPUT

```
  Serial.begin(9600);
```

- Starts serial communication at 9600 baud rate so you can see the sensor readings on the Serial Monitor (in the Arduino IDE).

```
  lcd.init();
  lcd.backlight();
}
```

- lcd.init(): Initializes the LCD screen.
- lcd.backlight(): Turns on the LCD's backlight for visibility.

7.5. Distance Measuring Function

```
  void ultrasonic () {
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);
```

- This sequence sends a 10 microsecond pulse from the trig pin.
- This triggers the ultrasonic sensor to send an ultrasonic sound wave.

```
  duration = pulseIn(echo, HIGH);
```

- pulseIn() waits for the echo pin to go HIGH and measures how long it stays HIGH.
- This gives the time it took for the ultrasonic pulse to return (round trip).

```
  distance = duration * 0.034 / 2;
}
```

- Converts time into distance in centimeters.
    - 0.034 cm/µs is the speed of sound (343 m/s at room temperature).
    - Divide by 2 because of the sound travels to the object and back.

7.6. The Loop Function

This function runs repeatedly forever.

```
void loop() {
  ultrasonic();
```

- Calls the function to update the latest distance.

If Object is Close (≤ 30 cm)

```
if (distance <= 30) {
  digitalWrite(led, LOW);
```

- If the object is 30 cm or closer, turn OFF the LED.

```
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Distance=");
  lcd.print(distance);
  lcd.print("cm");
```

- Clears the LCD and displays the actual distance on the first line.
- setCursor(0, 0): Moves to the first row, first column.

```
  lcd.setCursor(3, 1);
  lcd.print("LED is OFF!");
```

- Displays status message ("LED is OFF!") on the second row, a few spaces in for alignment.

```
  Serial.print("Distance is ");
  Serial.print(distance);
  Serial.print("cm.");
  Serial.println("LED is OFF!");
  delay(1000);
}
```

- Outputs the same info to the Serial Monitor, and waits for 1 second (1000 ms) before repeating the loop.

If Object is Farther Than 30 cm

```
else {
  digitalWrite(led, HIGH);
```

- If the object is farther than 30 cm, turn ON the LED.

```
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Distance is over");
  lcd.setCursor(3, 1);
  lcd.print("LED is ON!");
```

- Shows a warning or general status instead of showing the precise distance.
- Second row shows that the LED is ON.

```
  Serial.print("Distance is out of range. Distance is ");
  Serial.print(distance);
  Serial.print("cm.");
  Serial.println("LED is ON!");
  delay(1000);
}
```

- Prints a message to the Serial Monitor with the same idea.
- Waits for 1 second before checking the distance again.

7.7. Code Summary

setup()

- Sets up pin modes.
- Initializes Serial and LCD.

ultrasonic()

- Sends 10µs pulse via trig.
- Reads echo duration in µs.
- Converts to cm using formula:

  distance = duration × 0.034 / 2;

loop()

- Calls ultrasonic() to update distance.
- If distance ≤ 30, system:
  - Turns LED OFF
  - Displays distance
  - Logs "LED is OFF!"
- If distance > 30, system:
  - Turns LED ON
  - Displays "Distance is over"
  - Logs "LED is ON!"

---

8. Testing and Troubleshooting

8.1. Testing Procedures

To ensure that each subsystem functions correctly, the following testing steps should be followed in sequence:

8.1.1. Power-On Test

- **Objective**: Ensure the Arduino and all peripherals receive power.
- **Steps**:
  - Connect the Arduino via USB or external 5V supply.
  - Observe the power LED on the Arduino and LCD backlight turning ON.
  - If the LED or LCD remains off, check power and ground connections.

8.1.2. Serial Communication Test

- **Objective**: Confirm that the Arduino is transmitting data via USB.
- **Steps**:
  - Open Serial Monitor in Arduino IDE.
  - Set baud rate to 9600 (match Serial.begin(9600)).
  - Look for printed messages like "Distance is ..." or "LED is OFF/ON!".
  - If not working, verify USB connection and ensure correct COM port is selected.

8.1.3. Ultrasonic Sensor Test

- **Objective**: Confirm distance measurement functionality.
- **Steps**:
  - Place a flat object in front of the sensor.
  - Check for changing values in Serial Monitor or LCD.
  - Move the object closer or farther and observe changes.
  - If no distance updates:
    - Verify TRIG and ECHO pin wiring.
    - Check if pulseIn() returns a non-zero value.

8.1.4. LED Indicator Test

- **Objective**: Validate LED alert logic based on distance threshold.
- **Steps**:
  - Place object at <30 cm → LED should turn OFF.
  - Place object >30 cm → LED should turn **ON**.
  - Observe LCD or Serial output for corresponding message.
  - If LED does not change state:
    - Check LED polarity.
    - Measure output voltage on Arduino pin D7 using multimeter.

8.1.5. LCD Display Test

- **Objective**: Ensure correct distance and message display on LCD.
- **Steps**:
  - LCD should display distance in cm and LED status.
  - If LCD is blank:
    - Confirm SDA/SCL wiring (A4/A5 on Uno/Nano).
    - Recheck I2C address (default is 0x27, can be scanned using I2C scanner sketch).
    - Ensure lcd.init() and lcd.backlight() are in setup().

8.2. Troubleshooting Guide

Problem	Possible Causes	Suggested Fixes
LCD does not display text	Incorrect I2C address, wiring issue	Use an I2C scanner sketch to confirm the address, check A4/A5 connections
Serial Monitor shows no data	Baud rate mismatch, COM port not selected	Set Serial Monitor to 9600 baud, ensure correct COM port
Distance always shows 0 or large value	Sensor not triggered or echo not received	Check TRIG/ECHO pins and ensure object is within sensor range (2–400 cm)
LED not turning ON/OFF	Logic error, damaged LED, missing resistor	Verify pin assignment and logic, test LED with direct power
LCD turns ON but text is corrupted	Power instability, wrong contrast	Adjust potentiometer on I2C backpack (if available)
Arduino not recognized by PC	Driver issue or faulty USB cable	Try different cable or USB port, reinstall Arduino drivers

8.3. Best Practices During Testing

- Always test individual modules before integrating them.
- Use Serial.print() generously during development to monitor variable values.
- Keep jumper wires short and connections secure to avoid intermittent faults.
- Use a multimeter to verify pin voltages when debugging non-responsive hardware.
- Keep documentation of expected I/O values to validate against observed behavior.

8.4. Testing the Code on WOKWI

Here is link to test code.

https://wokwi.com/projects/429806479526798337

9. Observation

During the development and testing of the Arduino-based ultrasonic distance measurement system, several key observations were made regarding its performance, behavior, and reliability:

9.1. Sensor Accuracy and Response

- The HC-SR04 ultrasonic sensor consistently provided accurate distance readings within its effective range (2 cm to ~400 cm), with an average measurement error of ±1–2 cm in ideal conditions.
- Accuracy was affected by surface type and angle of incidence flat, perpendicular surfaces yielded the most reliable echoes.
- Intermittent false readings occurred when measuring soft or angled surfaces, which absorb or deflect sound waves.

9.2. LED Behavior

- The LED response was immediate and clearly indicated whether the object was within or beyond the 30 cm threshold.
- The binary logic (ON/OFF) worked effectively for proximity-based alerts, making it suitable for basic safety or monitoring applications.

9.3. LCD Display Output

- The **I2C LCD** reliably displayed real-time data with minimal latency.
- Rapid updates using lcd.clear() caused flickering, but this was resolved by optimizing update timing and limiting unnecessary clears.
- Display readability was good in various lighting conditions, especially with the backlight enabled.

9.4. Serial Monitor Feedback

- The Serial Monitor provided valuable debug information, confirming that measured values matched physical changes in object distance.
- It was helpful for tracking behavior during dynamic testing and verifying logic during development.

9.5. Overall System Stability

- The system remained stable during continuous operation over extended periods (1+ hours), with no resets or crashes.
- All components performed reliably as long as connections remained secure and powered appropriately.

10. Challenges Faced

During the design, assembly, and testing of the Arduino-based distance measurement system, several technical and practical challenges were encountered:

10.1. Sensor Inaccuracy and Interference

- **Irregular Distance Readings**: Initially, the HC-SR04 sensor produced fluctuating distance values due to environmental factors such as ambient noise, object shape, and reflective surface angles.
- **Echo Signal Issues**: In some cases, pulseIn() failed to return stable values, especially when the target was soft (e.g., fabric) or at a steep angle. This required validation of sensor positioning and surface reflectivity.

10.2. LCD Display Flickering

- **Unnecessary lcd.clear() Calls**: Frequent use of lcd.clear() caused noticeable flickering and display instability.
- **Display Latency**: Updating the screen too frequently led to slow refresh rates, making the system appear unresponsive.

10.3. Wiring and Pin Configuration

- **Incorrect I2C Address**: The LCD display did not work initially due to an incorrect I2C address. This was resolved by using an I2C scanner sketch to identify the correct address (typically 0x27 or 0x3F).
- **Loose Connections**: Jumper wires on the breadboard occasionally became loose, resulting in inconsistent LED operation or sensor failure. This highlighted the importance of firm wiring, especially on breadboards.

10.4. LED Indicator Logic Issues

- **Threshold Edge Cases**: Rapid switching of the LED near the 30 cm threshold created unstable visual feedback. This was mitigated by introducing a small delay and considering future use of hysteresis to stabilize state transitions.

10.5. Power Supply Noise

- **USB Power Instability**: When powering the system through a laptop USB port, voltage instability occasionally caused flickering LCD backlights or false distance readings. This was solved by using a more stable 5V external power source.

10.6. Environmental and Testing Limitations

- **Limited Testing Environment**: Indoor environments with sound absorbing materials or obstacles affected measurement consistency.

- **Lack of Shielding**: Electromagnetic interference from nearby devices occasionally caused erratic sensor values, suggesting future improvements could include shielding or signal filtering.

---

11. Possible Enhancements

While the current system performs effectively for basic distance monitoring and LED alert functionality, several enhancements could significantly improve its performance, usability, and versatility in real world applications:

11.1. Sensor Accuracy and Stability

- **Noise Filtering**: Implement a moving average filter or median filter on the distance readings to reduce random fluctuations caused by environmental noise or reflective interference.
- **Multiple Readings per Cycle**: Take multiple measurements and average them for improved accuracy and reliability.

11.2. User Interface Improvements

- **Dynamic Threshold Setting**: Add buttons or a rotary encoder to let users adjust the 30 cm threshold in real time, making the system adaptable for different applications.
- **Enhanced Display**: Use a 20x4 LCD or an OLED screen to provide additional data, such as min/max distances, timestamps, or system status messages.

11.3. Power and Efficiency Upgrades

- **Sleep Mode Implementation**: Utilize the Arduino's low power modes during idle states to conserve energy, particularly for battery-powered deployments.
- **Backlight Control**: Automatically turn off the LCD backlight after a period of inactivity to reduce power draw.

11.4. Connectivity and IoT Integration

- **Wireless Communication**:
  - Add Bluetooth (HC-05) for remote monitoring via smartphone.
  - Integrate Wi-Fi (ESP8266/ESP32) to send distance data to a web server or IoT dashboard.
- **Data Logging**:
  - Connect an SD card module to store distance logs over time.
  - Include timestamps to allow for historical trend analysis.

11.5. Alert and Control Features

- **Multimodal Alerts**:

    - Add a buzzer or vibration motor for audible or tactile warnings.
    - Use RGB LEDs to indicate distance ranges (e.g., green = safe, red = alert).
- **Actuator Control**:
    - Integrate servos or motors for automated doors or barriers triggered by detected distance.

11.6. Expandability

- **Multi-Sensor Setup**: Incorporate multiple ultrasonic sensors for 360° object detection or coverage of multiple zones.
- **Weather-Proofing**: Encapsulate the circuit in a protective housing for outdoor use.

---

12. Conclusion

This project successfully demonstrates the design and implementation of a distance-based alert system utilizing an Arduino microcontroller, HC-SR04 ultrasonic sensor, I2C LCD display, and an LED indicator. The primary objective to detect objects within a predefined proximity range and provide clear visual and serial feedback has been met with reliable and repeatable performance.

The system works by measuring the time-of-flight of ultrasonic pulses using the HC-SR04 sensor, converting the time data into distance using a simple physics-based formula that accounts for the speed of sound in air. The microcontroller processes this information and compares it to a fixed threshold of 30 centimeters. Based on this comparison:

- If the object is closer than or equal to 30 cm, the LED turns OFF and the LCD warns the user.
- If the object is farther than 30 cm, the LED remains **ON** and a safe-distance message is shown.

The LCD display provides user friendly, real time information with minimal power and I/O pin usage due to its I2C interface, freeing up the Arduino's digital pins for additional functionality if needed. Simultaneously, the Serial Monitor output offers detailed logs, which are valuable for debugging, calibration, or future integration with data logging systems.

---

13. References

1. **Arduino Documentation**
2. **HC-SR04 Sensor Datasheet**
3. **LiquidCrystal_I2C Library**
4. **Wokwi Arduino Simulator**
