```
#include <Servo.h>
#include <LiquidCrystal.h>

// Pin definitions Ulrasonic
const int trigPin = 10;
const int echoPin = 11;

// Pin definitions Buzzer
const int ledPin = 4;
const int buzzerPin = 5;

// Pin definitions Servo
const int servoPin = 3;

// Pin definitions RGB
const int redPin = 9;
const int greenPin = 10;

// Define the ultrasonic sensor range
const int distanceThreshold = 50; // Distance in cm to trigger the gate

// Initialize LCD (non-I2C)
LiquidCrystal lcd(A4, A5, A0, A1, A2, A3);

// Create a Servo object
Servo gateServo;

// Variables for ultrasonic sensor
long duration;
int distance;

void setup() {
```

```
  // Set up pins
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(buzzerPin, OUTPUT);
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);

  // Initialize the servo at closed position (0 degrees)
  gateServo.attach(servoPin);
  gateServo.write(0);  // Gate closed

  // Initialize the LCD
  lcd.begin(16, 2);
  lcd.print("Smart Gate Ready");

  // Set RGB to Red (Gate closed)
  digitalWrite(redPin, HIGH);
  digitalWrite(greenPin, LOW);

  // Small delay to start
  delay(1000);
}

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

  // Trigger the ultrasonic pulse
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
```

```
  digitalWrite(trigPin, LOW);

  // Read the echo pin
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance
  distance = duration * 0.034 / 2;

  // Clear LCD and display distance
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Distance: ");
  lcd.print(distance);
  lcd.print(" cm");

  // Check if the object is within range
  if (distance > 0 && distance <= distanceThreshold) {
    lcd.setCursor(0, 1);
    lcd.print("Gate Opening...");

    // Turn on the green LED (Gate opening) and turn off the red
    digitalWrite(greenPin, HIGH);
    digitalWrite(redPin, LOW);

    // Activate Buzzer while gate is opening
    digitalWrite(buzzerPin, HIGH);

    // Open the gate (90 degrees)
    gateServo.write(90);

    // Keep the gate open for 2 seconds
    delay(2000);
```

```
    // Turn off the buzzer after the gate is fully open
    digitalWrite(buzzerPin, LOW);

    // Clear the LCD before closing the gate
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Gate Closing...");

    // Close the gate (0 degrees)
    gateServo.write(0);

    // Set RGB to Red (Gate closed) and turn off green
    digitalWrite(redPin, HIGH);
    digitalWrite(greenPin, LOW);

    delay(1000);  // Allow the gate to close
  } else {
    // Clear LCD and display message when no object is detected
    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print("No Object Found");
  }

  delay(500);  // Small delay between readings
}
```

END OF CODE

SEE THE NEXT PAGES FOR DETAIL EXPLANATION

Step-by-Step Explanation of the Arduino Code

---

1. Include Required Libraries

```
#include <Servo.h>
#include <LiquidCrystal.h>
```

- Servo.h lets you control servo motors.
- LiquidCrystal.h lets you control a non-I2C LCD (16x2 in this case).

---

2. Define Pins

```
const int trigPin = 10;
const int echoPin = 11;
const int ledPin = 4;
const int buzzerPin = 5;
const int servoPin = 3;
const int redPin = 9;
const int greenPin = 10;
```

- Assigns pins for **ultrasonic sensor**, **LEDs**, **buzzer**, **servo**, and **RGB LEDs**.

---

3. Set the Distance Threshold

```
const int distanceThreshold = 50;
```

- If an object is detected **within 50 cm**, the gate will open.

---

4. Initialize LCD & Servo

```
LiquidCrystal lcd(A4, A5, A0, A1, A2, A3);
Servo gateServo;
```

- Defines how the LCD is wired and creates a servo control object.

---

5. Declare Variables

```
long duration;
int distance;
```

- duration: time it takes for the ultrasonic pulse to return.
- distance: calculated in centimeters.

6. Setup Function

```
void setup() {
  ...
}
```

Inside setup:

- **Configure pin modes** (input/output).
- **Attach servo** to pin 3 and set it to 0° (gate closed).
- **Initialize LCD** and print welcome message.
- **Turn on red LED** to show gate is closed.
- **Small delay** to stabilize components.

---

7. Loop Function

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

This part keeps running forever. It handles:

a. Ultrasonic Sensor Trigger

```
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
```

- Sends a 10µs pulse to **trigger** the ultrasonic sensor.
- Measures how long it takes for the echo to return.

b. Calculate Distance

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

- Converts time into **distance in cm** (using speed of sound formula).

c. Display Distance

```
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Distance: ");
```

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

- Displays current object distance on the LCD.

d. If Object is Detected Within Range

```
if (distance > 0 && distance <= distanceThreshold) {
  ...
}
```

Actions taken:

- **Display "Gate Opening…"**
- **Turn on green LED** and **turn off red LED**
- **Sound buzzer**
- **Open gate** using gateServo.write(90)
- **Wait 2 seconds**
- **Turn off buzzer**
- **Display "Gate Closing…"**
- **Close gate** using gateServo.write(0)
- **Turn red LED back on**, turn green off

e. If No Object is Found

```
else {
  lcd.clear();
  lcd.setCursor(0, 1);
  lcd.print("No Object Found");
}
```

- Displays a message indicating no object is in range.

---

8. Delay Between Readings

```
delay(500);
```

- Adds a small delay to reduce sensor overuse and flickering.

GateGuardian An Ultrasonic-Smart Gate

☑ Summary

This project:

- Monitors distance using an ultrasonic sensor.
- Automatically opens a gate when something is nearby.
- Uses a buzzer, LEDs, servo motor, and LCD for interaction.
- It's a simple and efficient **smart entry control system**.