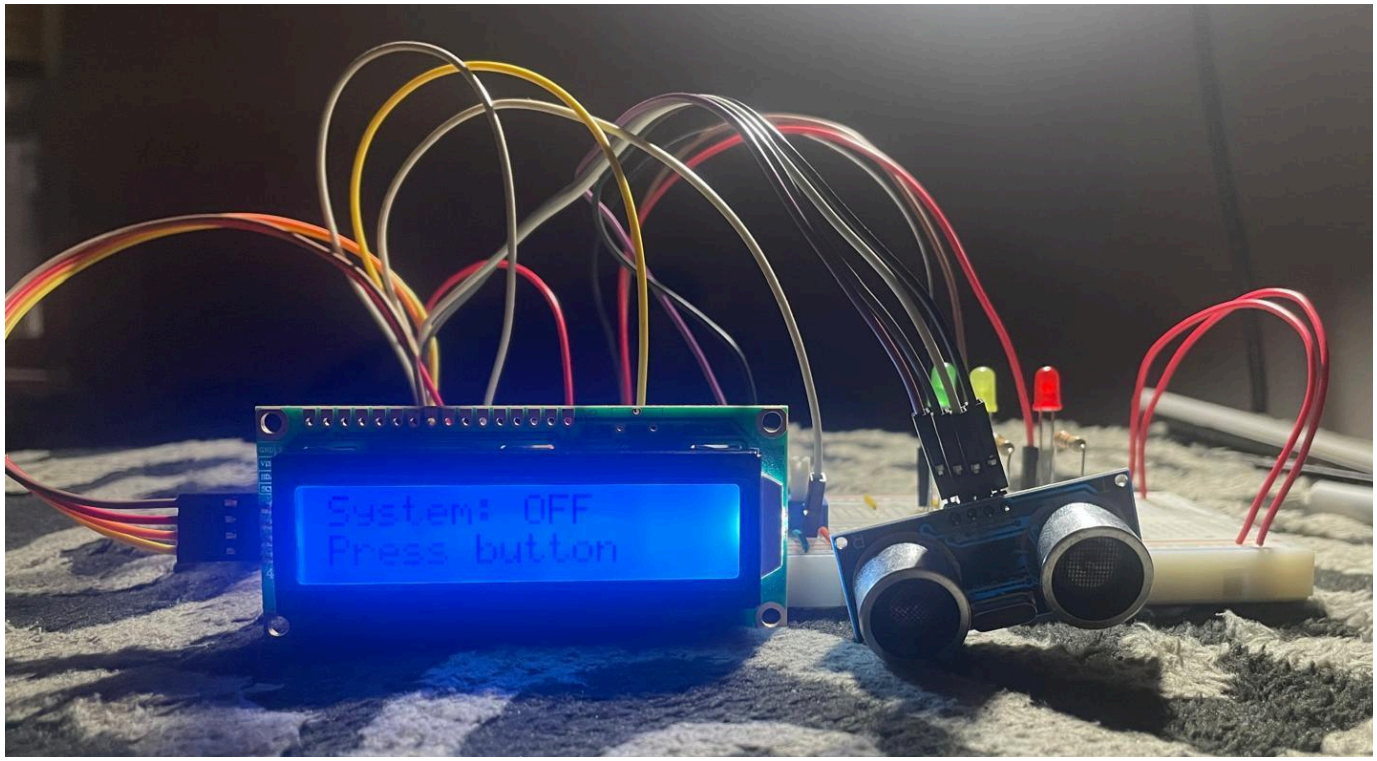
Title: Smart Object Detection with LCD



Author: B Hydara

Department: Dept. of Computer Eng. Tech.

Date: 2/10/2026

Table of Contents:

1. Objectives
2. Component List
3. Procedure

4. Diagrams and Pictures
5. Measurements
6. Source Code

7. Critical Thinking and Discussion
8. Conclusion
9. Appendix: Troubleshooting

1. Objectives:

- Learn the theory and operation of the ultrasonic sensor and LCD module
- Measure distance using an ultrasonic sensor and Arduino
- Display real-time distance measurements on an LCD screen
- Control LED indicators based on object distance
- Use a push button to enable/disable the system
- Use a potentiometer to control LED brightness
- Design a practical computer-controlled parking assistance system

2. Component / Equipment List:

- Arduino UNO with USB cable
- Breadboard and Jumper Wires
- Ultrasonic sensor
- 16x2 I2C LCD display
- 3 RGB LED with three 330 ohm resistors
- Push button and Potentiometer
- Multimeter for circuit testing and troubleshooting

3. Procedure:

Part 1: Sensor Testing

1. Connect the ultrasonic sensor to Arduino
2. Upload the distance measurement code
3. Verify distance values in Serial Monitor

Part 2: LED Distance Indicator

4. Add three LEDs to the circuit
5. Program LEDs to respond to distance ranges:
    - 2–8 inches: Red LED
    - 9–15 inches: Yellow LED
    - 16–22 inches: Green LED

Part 3: LCD Integration

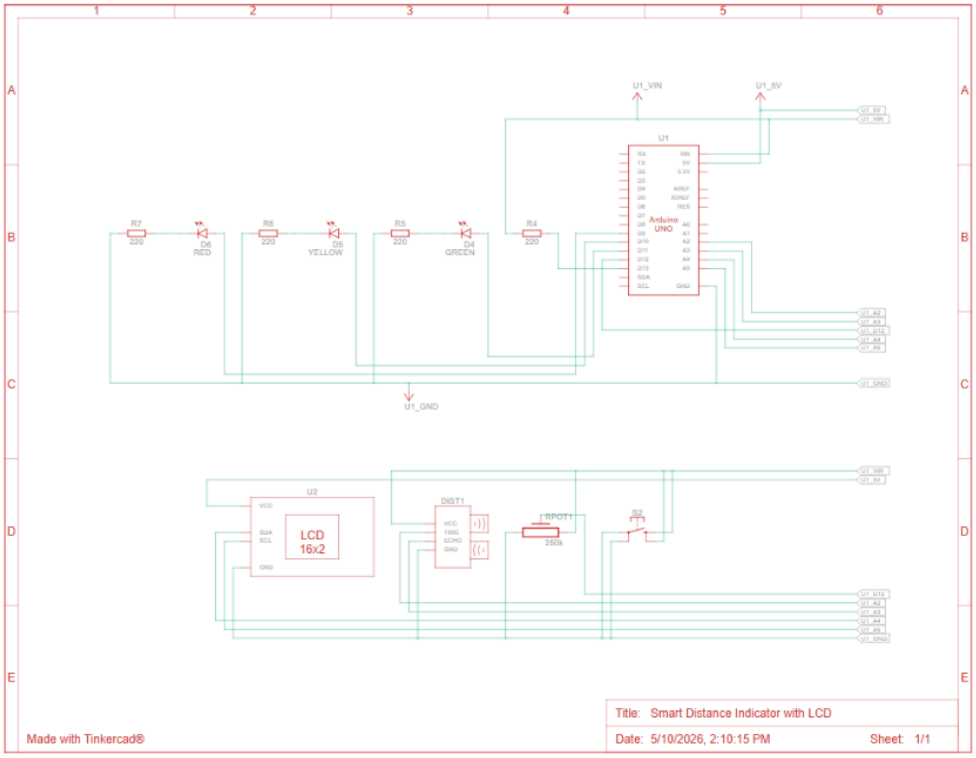
6. Connect the I2C LCD module using SDA and SCL pins
7. Display distance values and system status on the LCD

Part 4: Additional Controls

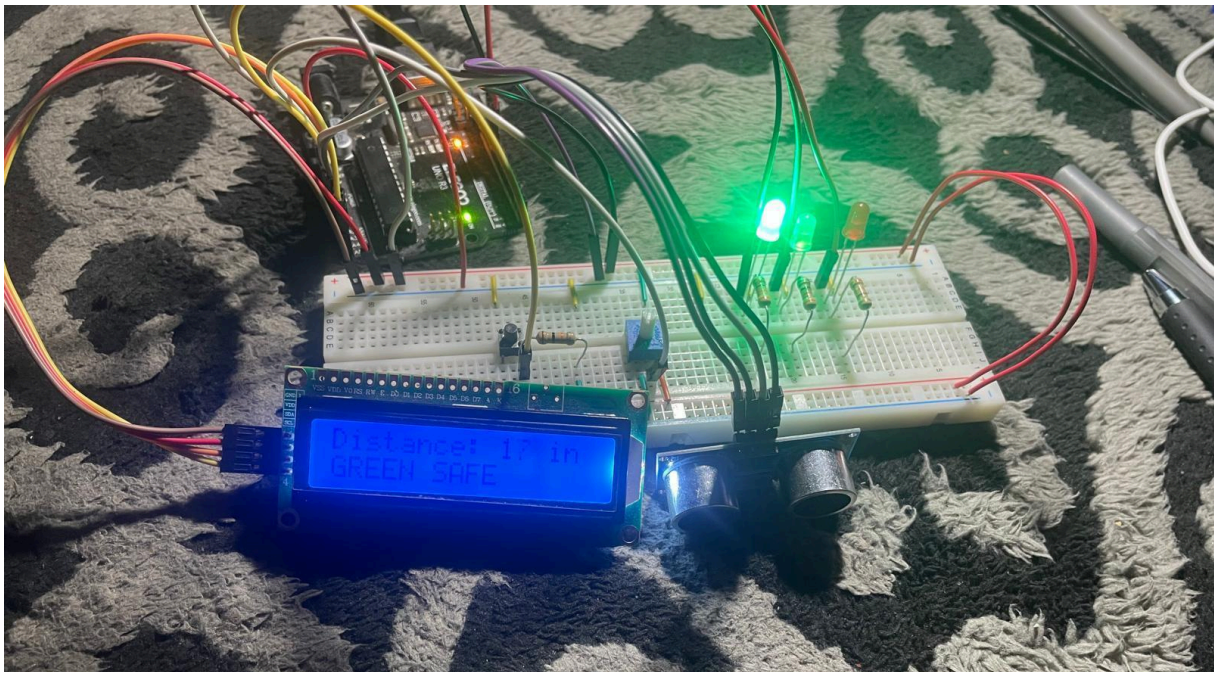
8. Add push button to turn system ON/OFF
9. Add potentiometer to control LED brightness
10. Test the complete system and verify all functions work correctly

4. Diagrams and Pictures:

Hardware (Schematic Diagram)

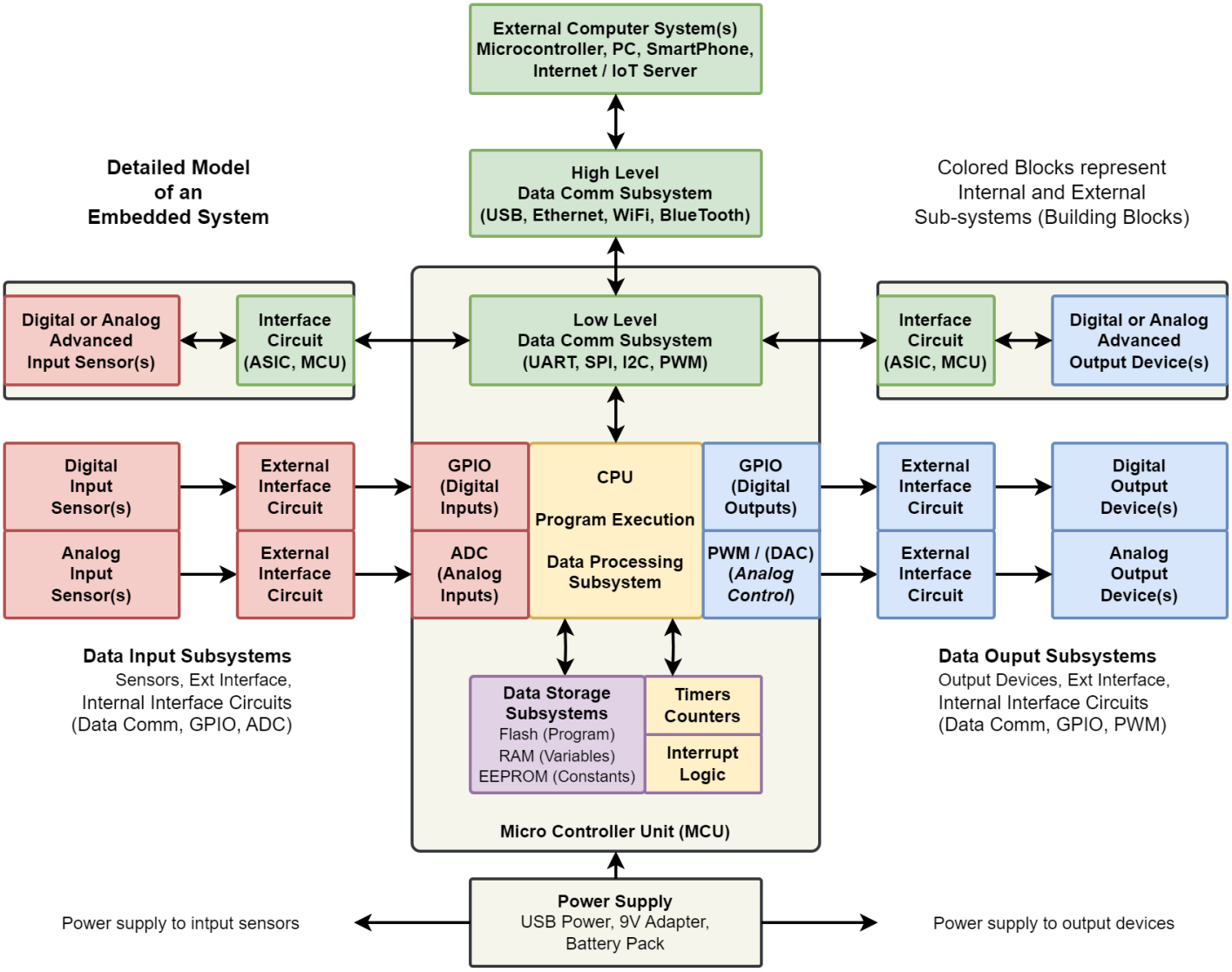


Hardware: (pictures of the final working version)



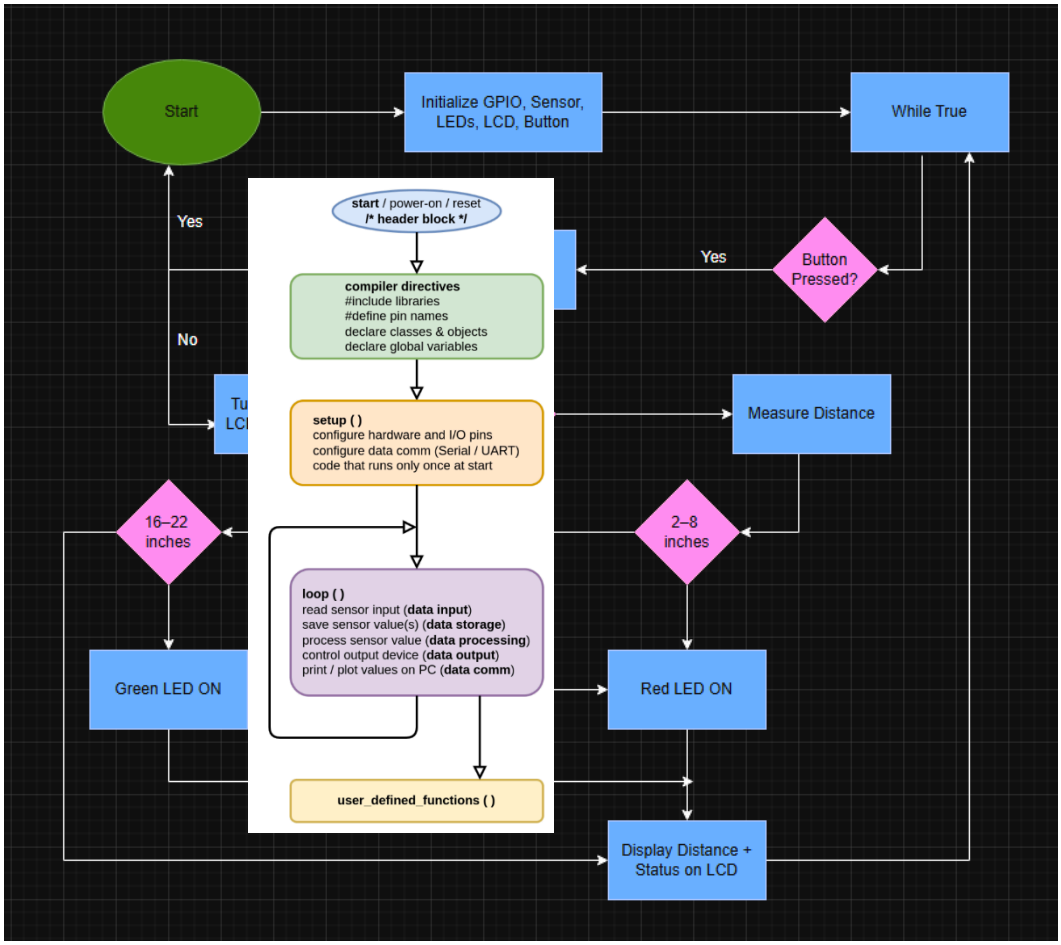
4. Diagrams and Pictures:

Hardware: (reference block diagram)

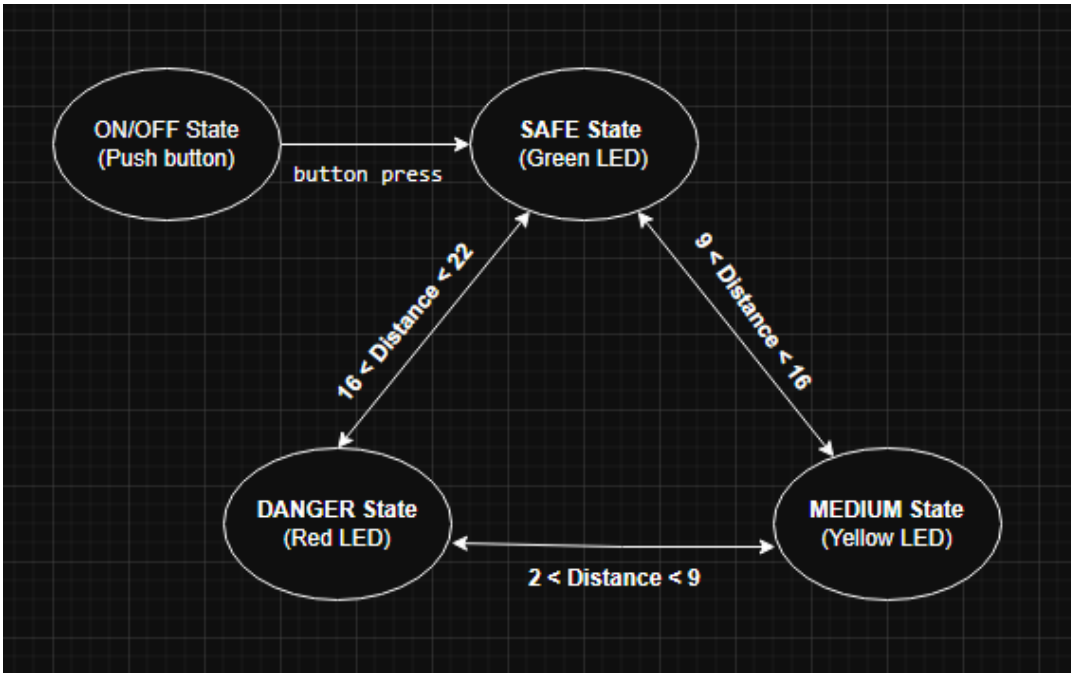


4. Diagrams and Pictures:

Software: (Flow chart)

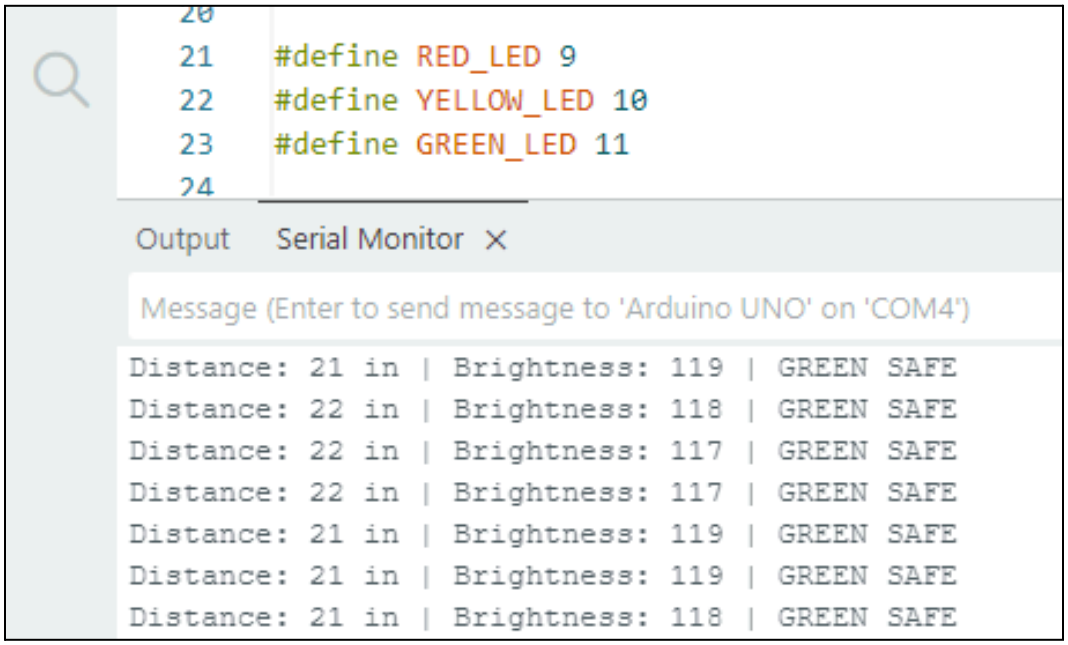


State diagram for program (FSM_RGB_LED)



5. Measurements:

- **Serial Monitor screenshots**



- **Serial Plotter screenshots and measurements**

6. Source Code:

```
/*
Name: Smart Distance Indicator with LCD.

This project uses an ultrasonic sensor and Arduino to measure distance and display the value
on an LCD screen.
Based on the measured distance, red, yellow, and green LEDs turn ON to indicate close,
medium, or safe distance ranges.
A push button is used to turn the system ON/OFF, and a potentiometer controls LED brightness.

Created by Balagie Hydara
Date: 5/10/2026

You can watch the video here: https://youtu.be/6sOthn_fJXs

*/

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

#define POT_PIN A0
#define BUTTON_PIN A1
#define TRIG_PIN A2
#define ECHO_PIN A3

#define RED_LED 9
#define YELLOW_LED 10
#define GREEN_LED 11

LiquidCrystal_I2C lcd(0x27, 16, 2);

bool systemOn = true;
bool lastButtonState = HIGH;

void setup() {
  Serial.begin(9600);

  pinMode(POT_PIN, INPUT);
  pinMode(BUTTON_PIN, INPUT_PULLUP);

  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  pinMode(RED_LED, OUTPUT);
  pinMode(YELLOW_LED, OUTPUT);
  pinMode(GREEN_LED, OUTPUT);

  lcd.init();
  lcd.backlight();

  lcd.setCursor(0, 0);
  lcd.print("Parking System");
  lcd.setCursor(0, 1);
  lcd.print("Starting...");
  delay(1500);
  lcd.clear();
```

```
}

void loop() {
  bool buttonState = digitalRead(BUTTON_PIN);

  // Push button turns system ON/OFF
  if (lastButtonState == HIGH && buttonState == LOW) {
    systemOn = !systemOn;
    delay(250);
  }

  lastButtonState = buttonState;

  if (systemOn == false) {
    allLEDsOff();

    lcd.setCursor(0, 0);
    lcd.print("System: OFF     ");
    lcd.setCursor(0, 1);
    lcd.print("Press button    ");

    Serial.println("System OFF");
    delay(300);
    return;
  }

  int potValue = analogRead(POT_PIN);
  int brightness = map(potValue, 0, 1023, 0, 255);

  long distance = getDistanceInches();
  distance = constrain(distance, 2, 22);

  allLEDsOff();

  String statusText = "";

  if (distance >= 2 && distance <= 8) {
    analogWrite(RED_LED, brightness);
    statusText = "RED CLOSE";
  }
  else if (distance >= 9 && distance <= 15) {
    analogWrite(YELLOW_LED, brightness);
    statusText = "YELLOW MID";
  }
  else if (distance >= 16 && distance <= 22) {
    analogWrite(GREEN_LED, brightness);
    statusText = "GREEN SAFE";
  }

  lcd.setCursor(0, 0);
  lcd.print("Distance: ");
  lcd.print(distance);
  lcd.print(" in  ");

  lcd.setCursor(0, 1);
  lcd.print(statusText);
  lcd.print("        ");
```

```
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.print(" in | Brightness: ");
  Serial.print(brightness);
  Serial.print(" | ");
  Serial.println(statusText);

  delay(300);
}

long getDistanceInches() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(5);

  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  long duration = pulseIn(ECHO_PIN, HIGH);

  long inches = duration / 74 / 2;
  return inches;
}

void allLEDsOff() {
  analogWrite(RED_LED, 0);
  analogWrite(YELLOW_LED, 0);
  analogWrite(GREEN_LED, 0);
}
```

7. Critical Thinking and Discussion:

Hardware Sub-systems: Description | Theory of Operation | Datasheets

Theory of Operation of Ultrasonic Sensor:

The ultrasonic sensor works by sending out a short burst of high-frequency sound waves (ultrasonic waves) and then listening for the echo that comes back after hitting an object. The Arduino sends a trigger pulse, and the sensor emits 8 cycles of 40 kHz sound. The sensor then sets the echo pin HIGH and waits until the sound reflects back. When the echo is received, the pin goes LOW. The time the echo pin stays HIGH represents the travel time of the sound wave. The Arduino measures this time and calculates the distance using the speed of sound.

Basic Specifications of the Ultrasonic Sensor:

- Operating Voltage: 3.3v ~ 5V DC
- Operating Current: less than 2 mA (static)
- Operating Frequency: 40 kHz
- Minimum Distance: about 2 cm
- Maximum Distance: up to 450 cm (about 4.5 m)
- Detection Angle: about 15°
- Sound Pressure: 112 dB
- Connector: 4-pin header with 2.54mm pitch
- VCC: 3.3v ~ 5V
- TRIG: Triggering Input Pin. 10uS TTL Pulses
- ECHO: TTL Logic Output Pin. Proportional to distance
- GND: Ground Pin

Micro-controller: Arduino Uno

- Clock: 16 MHz
- Flash: 32 KB
- RAM: 2 KB
- EEPROM: 1 KB

Pin	Function	Type	Description
1	NC	NC	Not connected
2	IOREF	IOREF	Reference for digital logic V - connected to 5V
3	Reset	Reset	Reset
4	+3V3	Power	+3V3 Power Rail
5	+5V	Power	+5V Power Rail
6	GND	Power	Ground
7	GND	Power	Ground
8	VIN	Power	Voltage Input
9	A0	Analog/GPIO	Analog input 0 / GPIO
10	A1	Analog/GPIO	Analog input 1 / GPIO
11	A2	Analog/GPIO	Analog input 2 / GPIO
12	A3	Analog/GPIO	Analog input 3 / GPIO
13	A4/SDA	Analog input/I2C	Analog input 4/I2C Data line
14	A5/SCL	Analog input/I2C	Analog input 5/I2C Clock line
-	-	DC Current per I/O Pin	20 mA
	-	Input Voltage (Nominal)	7–12V
-	-	EEPROM, Clock	1 KB, 16 MHz
-		Flash, RAM	32 KB, 2 KB

LED (Light Emitting Diode):

- Has two terminals:
  - Anode (+, long leg)
  - Cathode (−, short leg)
- Works only in one direction
- Requires a forward voltage to turn ON
- Produces light when current flows

Theory of operation:

When voltage is applied in the correct direction:

- Current flows from anode to cathode
- Electrons recombine inside the semiconductor
- Energy is released as light

If connected backward, no current flows, so LED stays OFF.

Practical tips:

- Always use a series resistor (220Ω–330Ω) to limit current
- Without a resistor to LED or Arduino pin can be damaged
- Arduino output pin gives about 5V, so resistor is needed to reduce current
- Typical safe current ≈ 20 mA or less

UART (Universal Asynchronous Receiver Transmitter)

- UART is the internal communication circuit inside the Arduino microcontroller.
- It is used for serial communication between Arduino and PC.

Key specifications:

- Asynchronous communication (no clock signal needed)
- Uses two lines:
  - TX (Transmit)
  - RX (Receive)
- Full duplex communication (send and receive at the same time)
- Baud rate used in this lab: **9600 bits per second**

How it works:

- Data is sent one bit at a time
- Each data frame includes:
  - Start bit
  - Data bits (usually 8 bits)
  - Stop bit

USB (Universal Serial Bus)

- USB is the external communication interface between Arduino and PC.
- It connects using a USB cable.

Key features:

- Converts USB signals to UART signals
- Provides both **data communication and power supply**
- Allows Arduino IDE Serial Monitor to display data

For more **Data Sheets Excerpts,** see the link below
UNO R3 | Arduino Documentation
PDF document

16 x 2 Character LCD

The LCD display Module is built in a LSI controller, the controller has two 8-bit registers, an instruction register (IR) and a data register (DR).

The IR stores instruction codes, such as display clear and cursor shift, and address information for display data RAM (DDRAM) and character generator (CGRAM). The IR can only be written from the MPU. The DR temporarily stores data to be written or read from DDRAM or CGRAM. When address information is written into the IR, then data is stored into the DR from DDRAM or CGRAM. By the register selector (RS) signal, these two registers can be selected.

General Specification

Item	Dimension	Unit
Number of Characters	16 characters x 2 Lines	—
Module dimension	80.0 x 36.0 x 13.2(MAX)	mm
View area	66.0 x 16.0	mm
Active area	56.2 x 11.5	mm
Dot size	0.55 x 0.65	mm
Dot pitch	0.60 x 0.70	mm
Character size	2.95 x 5.55	mm
Character pitch	3.55 x 5.95	mm
LCD type	FSTN Positive, Transflective	
Duty	1/16	
View direction	6 o'clock	
Backlight Type	LED, Triple-color	

Interface Pin Function

Pin No.	Symbol	Level	Description
1	V_{SS}	0V	Ground
2	V_{DD}	5.0V	Supply Voltage for logic
3	VO	(Variable)	Operating voltage for LCD
4	RS	H/L	H: DATA, L: Instruction code
5	R/W	H/L	H: Read(MPU→Module) L: Write(MPU→Module)
6	E	H,H→L	Chip enable signal
7	DB0	H/L	Data bus line
8	DB1	H/L	Data bus line
9	DB2	H/L	Data bus line
10	DB3	H/L	Data bus line
11	DB4	H/L	Data bus line
12	DB5	H/L	Data bus line
13	DB6	H/L	Data bus line
14	DB7	H/L	Data bus line
15	A	—	Supply power for LED +
16	R	—	Supply power for Red -
17	G		Supply power for Green -
18	B		Supply power for Blue -

potentiometer

Potentiometers also known as POT, are nothing but variable resistors. They can provide a variable resistance by simply varying the knob on top of its head. It can be classified based on two main parameters. One is their Resistance (R-ohms) itself and the other is its Power (P-Watts) rating.
The value or resistance decides how much opposition it provides to the flow of current. The greater the resistor value the smaller the current will flow. Some standard values for a potentiometer are 500Ω, 1K, 2K, 5K, 10K, 22K, 47K, 50K, 100K, 220K, 470K, 500K, 1 M.
Resistors are also classified based on how much current it can allow; this is called Power (wattage) rating. The higher the power rating the bigger the resistor gets and it can also more current. For potentiometers the power rating is 0.3W and hence can be used only for low current circuits.

■Specification

Total Rotational Angle	300±5°
Maximum Operating Voltage	250V AC
Insulation Resistance	100MΩ at DC500V
Dielectric Strength	AC 500V for 1 minute
Rotational Torque	20-200 gf.cm
Rotational Stop Strength of the Shaft	6kgf for 60 sec
Rotation Life	15,000 cycles

Pin Configuration

Pin No.	Pin Name	Description
1	Fixed End	This end is connected to one end of the resistive track
2	Variable End	This end is connected to the wiper, to provide variable voltage
3	Fixed End	This end is connected to another end of the resistive track

Mechanical Specifications:
- Operating Life: 300,000 cycles min.
- Operating Temperature: -25°C to +70°C

Electrical Specifications:
- Rating: 50mA / 12VDC
- Contact Resistance: 100mΩ max.
- Insulation Resistance: 100MΩ min.
- Dielectric Strength: 250VAC for 1 minute

Push Button

Push Buttons are normally-open **tactile switches**. Push buttons allow us to power the circuit or make any particular connection only when we press the button. Simply, it makes the circuit connected when pressed and breaks when released. A push button is also used for triggering of the SCR by gate terminal. These are the most common buttons which we see in our daily life electronic equipment's. Some of the applications of the Push button are mentioned at the end of the article.

Features

- Prevent flux rise by the insert-molded terminal
- Snap-in mount terminal
- Contact Bounce: MAX 5mS
- Crisp clicking by tactile feedback
- Dielectric Withstanding Voltage 250V AC for 1 minute

Technical Specifications

- Mode of Operation: Tactile feedback
- Power Rating: MAX 50mA 24V DC
- Insulation Resistance: 100Mohm at 100v
- Operating Force: 2.55±0.69 N
- Contact Resistance: MAX 100mOhm
- Operating Temperature Range: -20 to +70 ℃
- Storage Temperature Range: -20 to +70 ℃

Software Sub-systems:

Describing library functions used in this project.

digitalWrite()

- **Syntax:** digitalWrite(pin, value);
- **Parameters:**
  pin = output pin number
  value = HIGH or LOW
- **Return value:** None
- **Description:** Turns a digital output ON or OFF.

analogWrite()

- **Syntax:** analogWrite(pin, value);
- **Parameters:**
  pin = PWM pin number
  value = brightness value (0–255)
- **Return value:** None
- **Description:** Controls LED brightness using PWM.

analogRead()

- **Syntax:** analogRead(pin);
- **Parameters:**
  pin = analog input pin
- **Return value:** Analog value (0–1023)
- **Description:** Reads analog value from potentiometer.

digitalRead()

- **Syntax:** digitalRead(pin);
- **Parameters:**
  pin = digital input pin
- **Return value:** HIGH or LOW
- **Description:** Reads push button state.

pulseIn()

- **Syntax:** pulseIn(pin, value);
- **Parameters:**
  pin = input pin
  value = HIGH or LOW
- **Return value:** Pulse duration in microseconds
- **Description:** Measures ultrasonic echo pulse time.

map()

- **Syntax:** map(value, fromLow, fromHigh, toLow, toHigh);
- **Parameters:**
  value = input value
  fromLow/fromHigh = input range
  toLow/toHigh = output range
- **Return value:** Mapped value
- **Description:** Converts one range of values into another range.

constrain()

- **Syntax:** constrain(x, a, b);
- **Parameters:**
  x = value to limit
  a = minimum value
  b = maximum value
- **Return value:** Constrained value
- **Description:** Limits a value within a specific range.

lcd.print()

- **Syntax:** lcd.print(data);
- **Parameters:**
  data = text or number
- **Return value:** Number of bytes written
- **Description:** Displays text or values on the LCD screen.

lcd.setCursor()

- **Syntax:** lcd.setCursor(column, row);
- **Parameters:**
  column = cursor position
  row = LCD row number
- **Return value:** None
- **Description:** Sets cursor position on LCD display.

8.. Conclusion:

- **Real-life application of this project:**

  A real-life example is a vehicle parking assistance system used in modern cars. The system uses ultrasonic sensors to detect nearby objects and provides feedback to the driver using lights, displays, or warning sounds to help avoid collisions while parking.

- **Some similarities and differences between the real-life system example and this project:**

  Similarities:
  - Both use ultrasonic sensors to measure distance
  - Both use a microcontroller/system to process sensor data
  - Both provide output feedback based on distance
  - Both are computer-controlled systems with input, processing, and output sub-systems

  Differences:
  - Real parking systems use more advanced sensors and higher accuracy
  - Real systems often include buzzers, cameras, and larger displays
  - The lab project uses simple LEDs and LCD for demonstration purposes
  - Real systems are designed for outdoor vehicle environments and longer distance ranges

9.. Appendix: Troubleshooting:

- Symptoms of the problem:
  - LCD not displaying text
  - LEDs not turning ON
  - System not responding to push button

- Hardware and/or software test performed to determine the cause of the problem:
  - Checked wiring connections
  - Verified LCD I2C address
  - Tested sensor readings in Serial Monitor
  - Tested LEDs individually

- Solution to fix the problem:
  - Corrected wiring issues
  - Changed LCD address to correct value
  - Fixed pin assignments in code
  - Verified power and ground connections