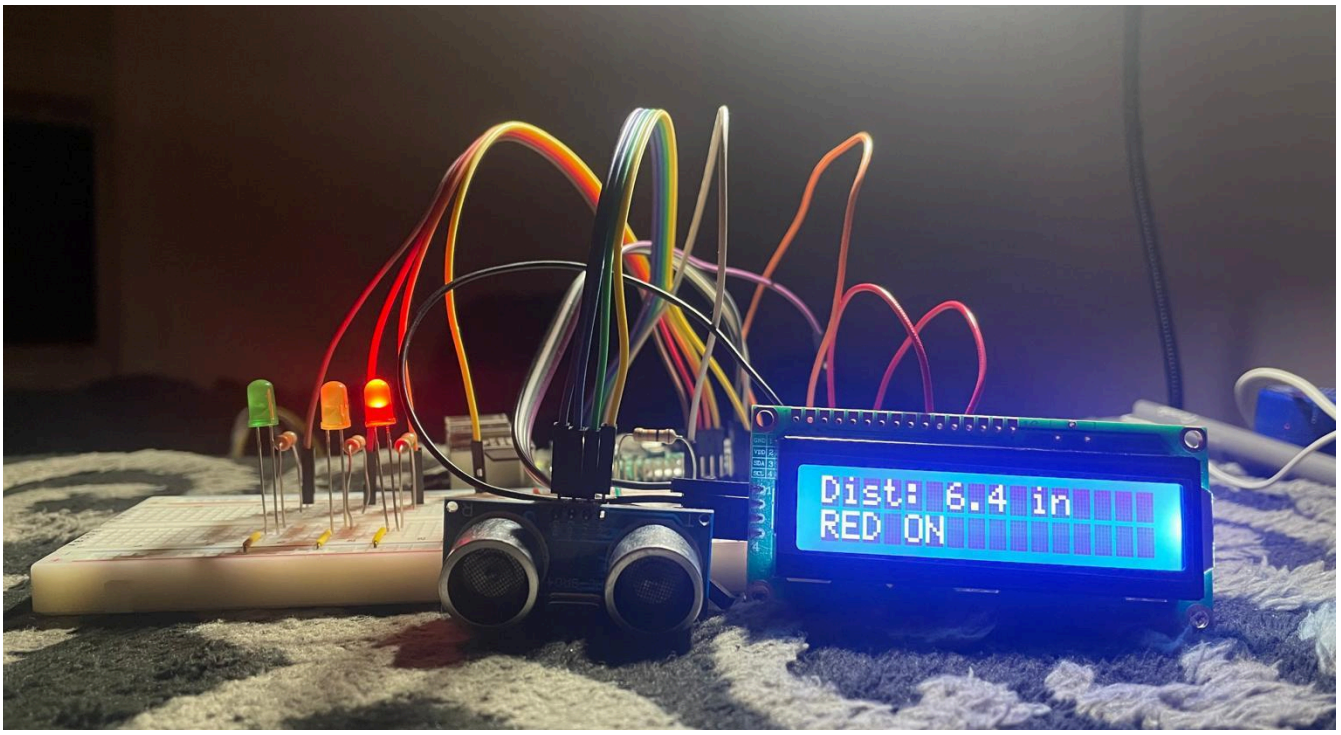
Project Title: Obstacle Detection in Raspberry Pi



Author: B Hydara

Department: Dept. of Computer Eng. Tech.

Date: 5/15/2026

Table of contents Page

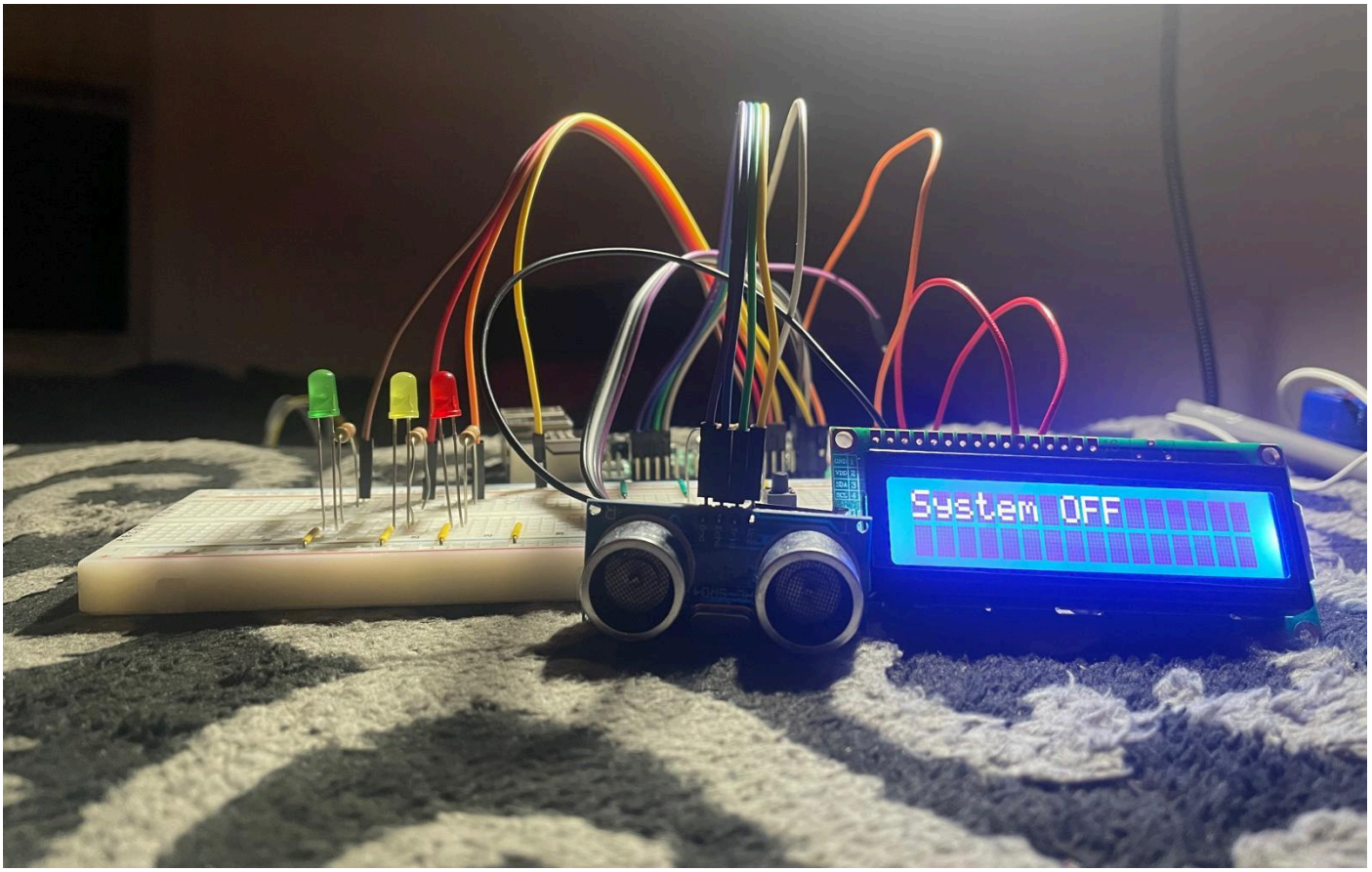
Introduction

This project demonstrates a smart obstacle detection system using a Raspberry Pi, ultrasonic sensor, LEDs, push button, and LCD display.

The system measures the distance between the sensor and nearby objects, displays real-time distance readings on the LCD, and provides visual feedback through colored LEDs.

A push button is used to enable or disable the system, creating an interactive embedded solution for distance monitoring and collision awareness.

The project simulates a real-world vehicle parking assistance system designed to help prevent collisions while parking.



Objectives

- Learn how to interface an ultrasonic sensor with Raspberry Pi
- Measure distance using Python and GPIO pins
- Display distance measurements on an LCD screen
- Control LEDs based on distance ranges
- Use a push button to enable and disable the system
- Design a practical computer-controlled parking assistance system

Equipment

- Raspberry Pi
- Ultrasonic Sensor
- 16x2 I2C LCD
- Red | Yellow | Green LEDs
- Push Button
- Breadboard & Jumper wires
- Resistors

Theoretical Background

The ultrasonic sensor works by transmitting high-frequency sound waves and measuring the time required for the echo to return after reflecting from an object. The Raspberry Pi calculates the distance using the speed of sound. Based on the measured distance, the program controls LEDs to indicate danger, medium, or safe distance ranges. The LCD display shows the real-time distance measurement and system status. The push button is used to enable or disable the system operation.

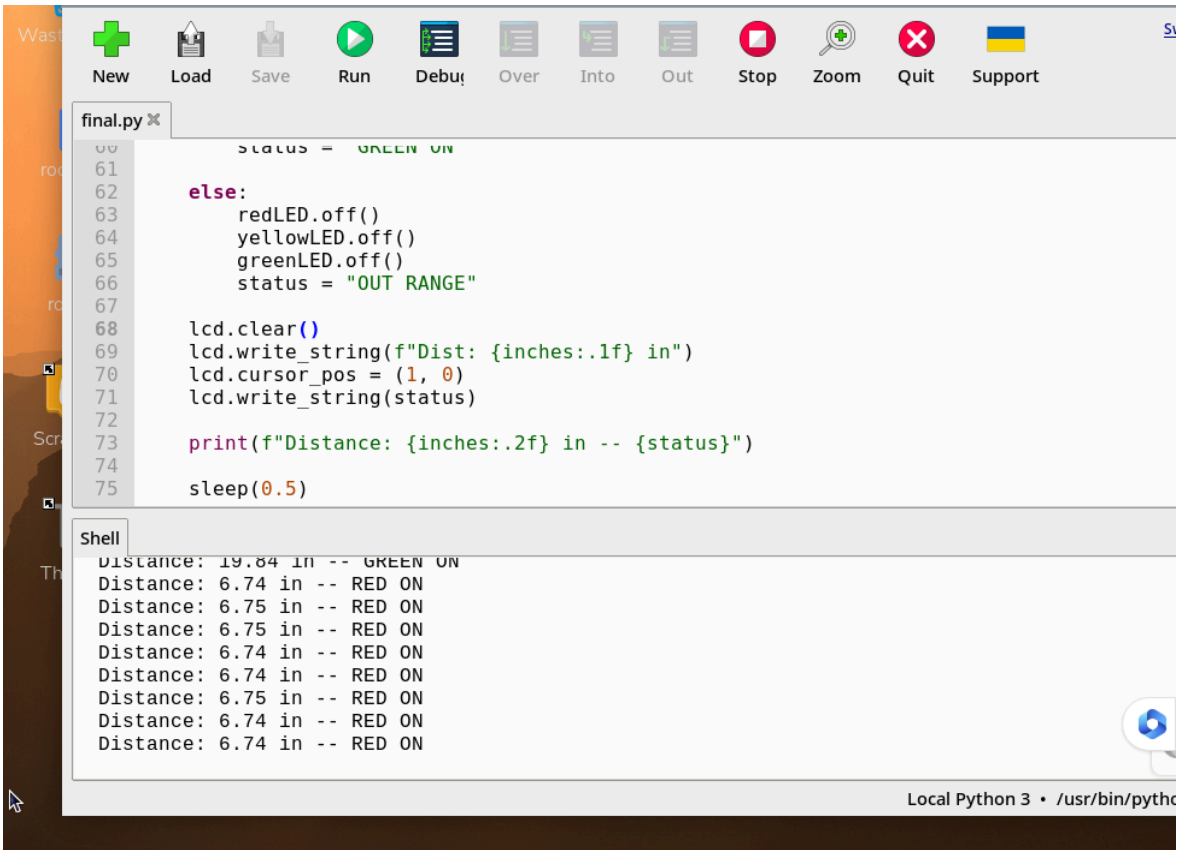
Procedure

1. Connect the ultrasonic sensor to the Raspberry Pi GPIO pins
2. Connect the red, yellow, and green LEDs with resistors
3. Connect the push button to GPIO pin 13
4. Connect the I2C LCD display to the Raspberry Pi SDA and SCL pins
5. Enable I2C communication using `raspi-config`
6. Install required Python libraries (`gpiozero`, `RPLCD`, `smbus2`)
7. Write and run the Python program in Thonny IDE
8. Test the system using different object distances
9. Verify that LEDs and LCD respond correctly

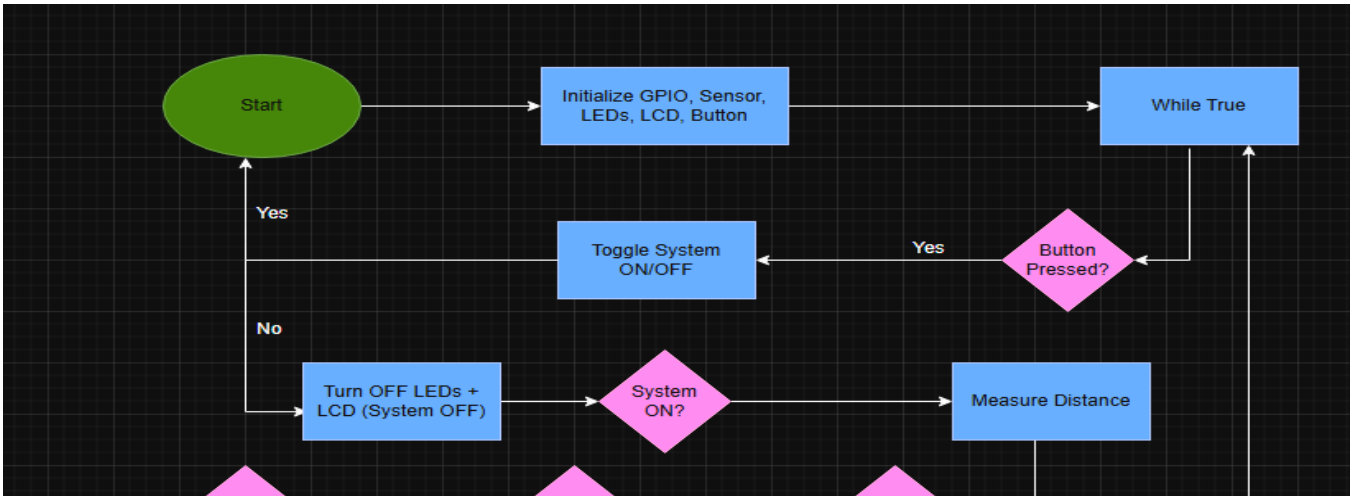
Data & Results

- Distance range 2–8 inches: Red LED turned ON
- Distance range 9–15 inches: Yellow LED turned ON
- Distance range 16–22 inches: Green LED turned ON
- LCD successfully displayed real-time distance values and LED status
- Push button correctly enabled and disabled the system
- Terminal window displayed distance measurements and system messages

Python Terminal output



Flowchart



Conclusion & Reflection

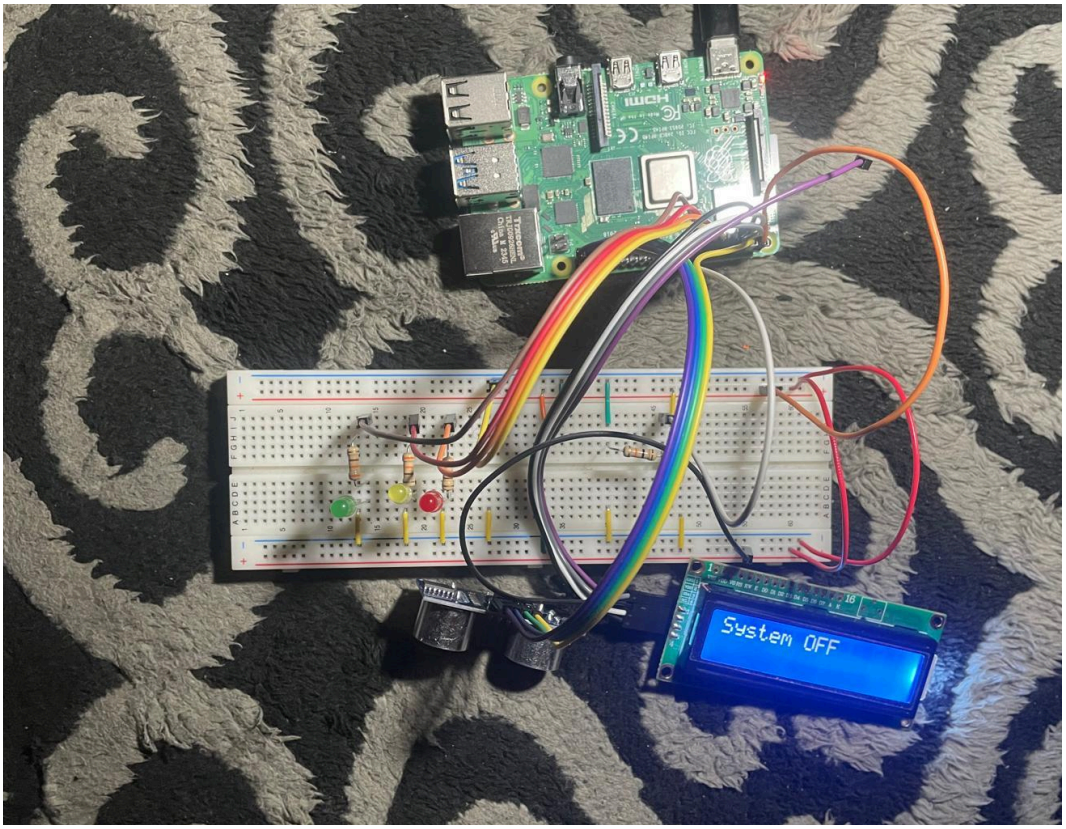
The Ultrasonic Parking System project successfully measured distance using an ultrasonic sensor and displayed the results using LEDs and an LCD screen. The project demonstrated how Raspberry Pi GPIO pins, Python programming, sensors, and output devices can be combined to create a practical computer-controlled system. One challenge during the project was configuring I2C communication for the LCD display, but the issue was solved by enabling I2C and installing the required libraries. This project improved understanding of Raspberry Pi hardware interfacing, GPIO control, and sensor-based automation systems.

References

- Raspberrypi
- Brightspace notes.
- Full video here: Smart Distance Indicator using Raspberry Pi

Appendix A (Screenshot)

Circuit setup



LCD output

Appendix B (Source code)

Python code

```
#Watch the video here: https://youtu.be/ieH1bMKlqek
from gpiozero import DistanceSensor, LED, Button
from RPLCD.i2c import CharLCD
from time import sleep

# Ultrasonic sensor
sensor = DistanceSensor(echo=26, trigger=19)

# LEDs
redLED = LED(17)
yellowLED = LED(27)
greenLED = LED(22)

# Push button
button = Button(13)

# LCD setup
lcd = CharLCD('PCF8574', 0x27)

systemOn = True
lastButtonState = False

while True:
    # Button toggle
    if button.is_pressed and not lastButtonState:
        systemOn = not systemOn
        lcd.clear()
        sleep(0.3)

    lastButtonState = button.is_pressed

    if not systemOn:
        redLED.off()
        yellowLED.off()
        greenLED.off()

        lcd.clear()
        lcd.write_string("System OFF")
        print("SYSTEM OFF")
        sleep(0.5)
        continue
```

```
    inches = sensor.distance * 100 / 2.54

    if 2 <= inches <= 8:
        redLED.on()
        yellowLED.off()
        greenLED.off()
        status = "RED ON"

    elif 9 <= inches <= 15:
        redLED.off()
        yellowLED.on()
        greenLED.off()
        status = "YELLOW ON"

    elif 16 <= inches <= 22:
        redLED.off()
        yellowLED.off()
        greenLED.on()
        status = "GREEN ON"

    else:
        redLED.off()
        yellowLED.off()
        greenLED.off()
        status = "OUT RANGE"

    lcd.clear()
    lcd.write_string(f"Dist: {inches:.1f} in")
    lcd.cursor_pos = (1, 0)
    lcd.write_string(status)

    print(f"Distance: {inches:.2f} in -- {status}")

    sleep(0.5)
```