

ULTRAGAUGE

GET DISTANCE DIGITALLY

(A Real-time Distance Measuring System Using ESP32)

PROJECT BY:

Abhishek Vishwakarma

BACHELOR OF ENGINEERING

(Computer Science & Engineering)

SESSION: 2024-2025

PROJECT COMPLETION CERTIFICATE

This is to certify that the project titled “**ULTRAGAUGE**” has been personally developed and completed by **Abhishek Kumar Vishwakarma**, B.Tech student, as a self-initiated learning and development project.

The work presented is original and not submitted as part of any academic curriculum. All development, coding, and testing were carried out independently for skill enhancement and personal growth.

Date: 26/05/2025

Signature:

(Abhishek Kumar Vishwakarma)

ABSTRACT

The project titled “**Ultragaugue – Get Distance Digitally**” presents a compact, cost-effective, and reliable distance monitoring system inspired by modern smart car technologies. Designed and developed using an **ESP32 microcontroller**, the system employs an **HC-SR04 ultrasonic sensor** for precise distance measurement, an **I2C OLED display** for real-time output, and a **buzzer alert system** for critical proximity warnings. The purpose of this project is to demonstrate how digital sensors can prevent vehicle collisions in constrained parking or tight spaces — a major concern in urban driving environments.

At boot, the system provides an engaging **animated car sequence**, followed by a 3-step introduction screen showing a welcome message, project name, and a “Powered by Abhishek” tag. Once operational, the system continuously measures the distance of obstacles in front of the sensor. If the object comes **closer than 4cm**, the buzzer activates as a safety warning, simultaneously displaying the current distance and alert message on the OLED screen. Components such as **BC547 transistor**, **resistors**, **MB102 power module**, and a **breadboard** form the supporting circuit for stable power and signal regulation.

The entire setup mimics how smart cars use proximity sensors to reduce chances of scratches, bumps, or minor crashes. It is modular and can be extended to support battery packs for portability or upgraded for wireless communication. This project not only strengthens concepts of embedded systems and sensor interfacing but also reflects real-world applications in smart automobile design and safety enhancement.

ACKNOWLEDGEMENT

I would like to take this opportunity to acknowledge the personal journey and motivation behind the successful completion of my project titled “**Ultragaugue – Get Distance Digitally.**” This project has been entirely conceptualized, designed, and implemented by me, driven by my curiosity and passion to explore real-world embedded technologies.

The idea, design, coding, circuit integration, and testing have been executed independently, with the sole aim of improving my practical skills and building a foundation for future work in IoT, smart systems, and automotive technologies.

I undertook this project as a self-initiated challenge to understand how proximity sensors, microcontrollers, and real-time feedback systems work together in modern smart vehicles. It gave me hands-on experience in circuit design, code structuring, debugging, and system logic building.

I am thankful to several open-source platforms, developer forums, and online communities, whose content and examples helped guide me through complex technical hurdles. I also thank my friends and family for their support and encouragement throughout the development process.

This project represents my personal effort to turn ideas into action — to not just learn but to **create**. It is a small yet meaningful step in my journey to grow as a technology enthusiast and developer.

Date: 26/05/2025

Abhishek Vishwakarma

Place: Lucknow

202310101150305

TABLE OF CONTENT

- I) TITLE
- II) CERTIFICATE
- III) ABSTRACT
- IV) ACKNOWLEDGEMENT
- V) TABLE OF CONTENT
- VI) LIST OF ABBREVIATIONS
- VII) REQUIREMENTS
 - 1. Software requirements
 - 2. Hardware requirements

VIII) CHAPTERS

1.0 Chapter 1: Introduction

- 1.1 Objective of the Project
- 1.2 Motivation and Purpose
- 1.3 Scope of the Project
- 1.4 Applications

2.0 Chapter 2: Literature Review

- 2.1 Existing Systems and Limitations
- 2.2 Research Gap and Innovation
- 2.3 Demonstration of project and its working in car parking system

3.0 Chapter 3: System Overview

- 3.1 Block Diagram
- 3.2 System Architecture
- 3.3 Hardware and Software Summary

4.0 Chapter 4: Components Used

- 4.1 ESP32 WROOM32 DevKit V1
- 4.2 HC-SR04 Ultrasonic sensor
- 4.3 Tactile Push Button Switch
- 4.4 OLED Display

4.5 Buzzer

4.6 Other Supporting Components

5.0 Chapter 5: Circuit Design and Implementation

5.1 Circuit Diagram

5.2 Pin Configuration

5.3 Hardware Connections

6.0 Chapter 6: Software and Coding

6.1 Tools and IDE Used

6.2 Code Structure

6.3 Animation Implementation

6.4 Distance Measurement Logic

6.5 Challenges Faced in Coding

7.0 Chapter 7: Working and Output

7.1 Boot and Start up Process

7.2 Real-time Monitoring

7.3 Crash Alert Scenario

7.4 Sample Outputs (Images/Descriptions)

8.0 Chapter 8: Conclusion and Future Scope

8.1 Summary of the Project

8.2 Limitations

8.3 Future Enhancements

9.0 Chapter 9: References

9.1 Web Links

9.2 Libraries Used

9.3 Tutorials/Communities Referred

10.0 Appendix

A.1 Complete Source Code

A.2 Additional Images

A.3 Pin Mapping Table

LIST OF ABBREVIATION

Abbreviation	Full Form
IoT	Internet of Things
ESP32	Espressif Systems Protocol 32-bit Microcontroller
MCU	Microcontroller Unit
OLED	Organic Light Emitting Diode
I2C	Inter-Integrated Circuit
HC-SR04	High-Accuracy Sonic Rangefinder 400cm
VCC	Voltage Common Collector (Power Supply Pin)
GND	Ground
GPIO	General Purpose Input/Output
IDE	Integrated Development Environment
GIF	Graphics Interchange Format
PWM	Pulse Width Modulation
USB	Universal Serial Bus
LED	Light Emitting Diode

REQUIREMENTS

Hardware Requirements

S.No	Component	Quantity	Description
1.	ESP32 WROOM32 DevKit V1	1	Main microcontroller board with Wi-Fi and GPIO support
2.	HC-SR04	1	For Ultrasonic wave send and capture
4.	1.3" OLED Display (I2C)	1	For visual feedback and animations
5.	Buzzer	1	For sound alerts in case of fire detection
6.	Resistors (as needed)	—	For signal conditioning (if required)
7.	Breadboard / PCB	1	For connecting components
8.	Jumper Wires	As needed	For wiring and connectivity
9.	Micro USB Cable	1	For power and programming
10.	Power Supply (5V via USB)	1	To power the ESP32 board

Software Requirements

S.No	Tool / Software	Purpose
1.	VS Code	Writing, compiling, and uploading code
2.	ESP32 Board Package	ESP32 support in Arduino IDE
3.	U8G2 Library	OLED graphics library for SH1106
4.	GIF to Frame Converter Tool	For converting GIFs into displayable frames
5.	Serial Monitor (built-in)	For debugging and output monitoring
6.	Image2cpp converter	To convert the images into hex code.

Chapter 1: Introduction

1.1 Objective of the Project

The primary objective of the *Ultragaugue* project is to develop a compact and cost-effective smart distance monitoring system using ultrasonic sensing technology. The device aims to assist in real-time proximity measurement and alert mechanisms to prevent collisions, especially in congested or tight spaces such as parking areas or garages.

1.2 Motivation and Purpose

With the growing use of smart systems in vehicles, even basic-level safety mechanisms are becoming increasingly important. Inspired by smart parking assist systems in modern cars, this project was initiated to create a similar concept using simple electronics and embedded systems. The motivation was to design an affordable prototype that demonstrates how smart sensing and visual feedback can enhance safety and awareness in tight driving environments.

1.3 Scope of the Project

The project focuses on:

- Real-time distance measurement using the HC-SR04 ultrasonic sensor.
- OLED-based display of visual feedback including animation and alerts.
- Generating buzzer alerts when distance goes below a threshold.
- Boot animations for an interactive user experience.
- Future scalability such as battery portability or integration with larger IoT systems. This prototype serves as a demonstration model and can be further developed for automotive or robotics applications.

1.4 Applications

- **Smart Car Parking Assist Systems:** Helps in preventing collisions with walls or other objects during reverse parking.
- **Obstacle Detection in Robotics:** Can be integrated into robots to detect nearby obstacles.
- **Industrial Automation:** Useful in conveyor systems to monitor distances between packages or machines.
- **Smart Garage Assist:** Alerts users when their vehicle is too close to the wall during parking.

Chapter 2: Literature Review

2.1 Existing Systems and Limitations

Various proximity and parking assistance systems are available in modern vehicles. Most of them rely on ultrasonic sensors or camera-based systems. These are generally expensive, require complex installation, and may not be feasible for small-scale projects or educational prototypes.

Common limitations in existing systems:

- High cost of commercial solutions
- Complex integration with vehicle systems
- Lack of customizable features for DIY use
- Power-hungry modules unsuitable for minimal hardware

2.2 Research Gaps and Innovation

Ultragaugue fills the gap by providing:

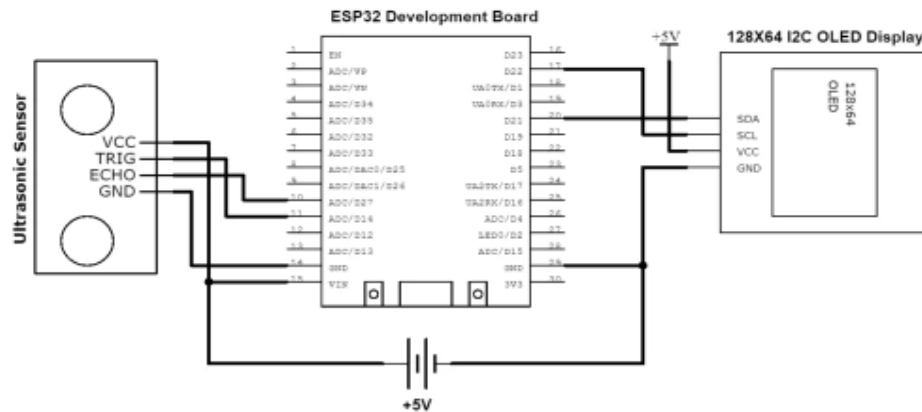
- A low-cost prototype for short-range obstacle detection
- Simple implementation using ESP32 and OLED
- Real-time distance measurement and alert system
- Educational value with hands-on experience in embedded systems

2.3 Demonstration of Project & Its Working in Car Parking Systems

- Ultragaugue uses an ultrasonic sensor (HC-SR04) to measure the real-time distance between the vehicle and nearby obstacles.
- The measured distance is displayed on an OLED screen, providing visual feedback to the driver during parking.
- When the distance becomes less than 4 cm, an audible buzzer alert warns the driver to prevent collision.
- This system effectively enhances parking safety by reducing manual errors and providing instant alerts.
- Ultragaugue fills this gap by integrating distance measurement, visual animation, and buzzer alerts into a single low-cost device.
- The innovation lies in combining sensor accuracy with interactive feedback to make parking safer and easier.

Chapter 3: System Overview

3.1 Block Diagram



- The ESP32 is the central controller that manages input from the ultrasonic sensor and button, processes the distance data, and provides visual and sound output via the OLED and buzzer.

3.2 System Architecture

The system follows a **modular architecture**, with each module (input, processing, and output) functioning independently but in sync:

- **Input Unit:**
 - **HC-SR04 Ultrasonic Sensor:** Sends ultrasonic pulses and receives echoes to calculate distance.
 - **Button:** Toggles display scale (cm, inch, feet, meter).
- **Processing Unit:**
 - **ESP32 DevKit V1:** Reads sensor data, performs calculations, handles logic for unit conversion and threshold checking, controls animations, and display.
- **Output Unit:**
 - **OLED Display:** Shows animated intro screens, real-time distance, alerts.
 - **Buzzer:** Alerts when object is closer than 4 cm (or equivalent in other units).

Chapter 4: Components Used

4.1 Hardware Components

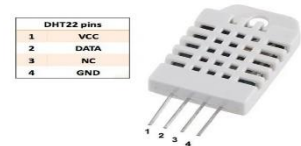
1. ESP32 WROOM32 DevKit V1

- Main microcontroller board based on ESP32 chip.
- Features Wi-Fi, Bluetooth, and multiple GPIO pins for sensor interfacing.
- Responsible for processing sensor data, controlling display, and generating alerts.



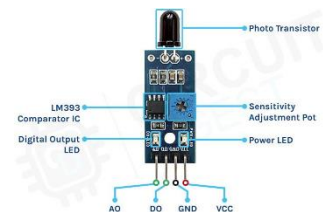
2. HC-SR04 Ultrasonic Sensor

- Measures distance by sending out ultrasonic waves and calculating time.
- Range: 2 cm to 400 cm.
- Accuracy: ~3 mm.
- Used to detect how far the car is from nearby objects.



3. Tactile Push Button Switch

- Used to toggle between different distance display units:
 - Centimeters (cm)
 - Inches (inch)
 - Feet (ft)
 - Meters (m)
- Debounced via software using a custom Button.h class.



4. OLED Display (1.3 inch, I2C interface)

- Small organic LED display for showing real-time temperature, humidity, and alert animations.
- Communicates with ESP32 via I2C protocol.
- Supports custom animations by displaying frames converted from GIFs.



5. Buzzer

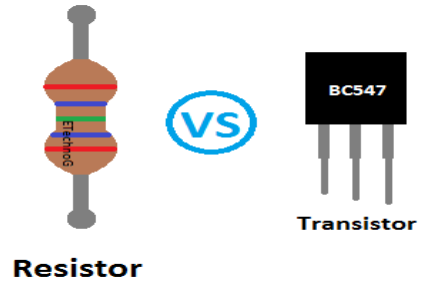
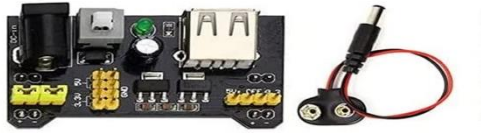
- Provides audible alerts when fire is detected.
- Controlled by ESP32's digital output pins.



6. Supporting Components

- Jumper wires for connections.

- Breadboard or PCB for mounting components.
- Resistors as needed for signal conditioning.
- Power supply via USB cable.



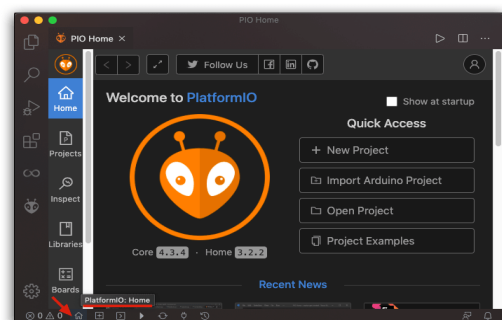
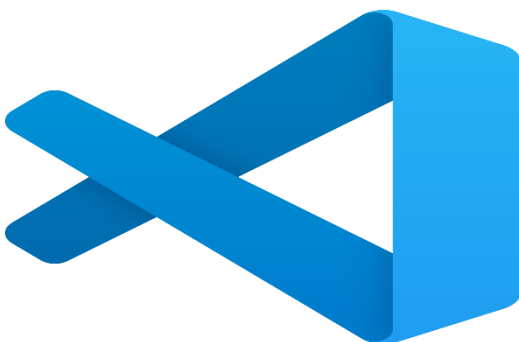
4.2 Software Components

1. Arduino IDE

- Development environment used for coding, compiling, and uploading firmware to ESP32.

2. Libraries Used

- **U8G2 library:** For interfacing OLED display .
- **Arduino library:** For controlling the hardware components.
- **Custom library:** Provides better enhancement to get precise code.
- Additional utility code for GIF frame handling and animation.

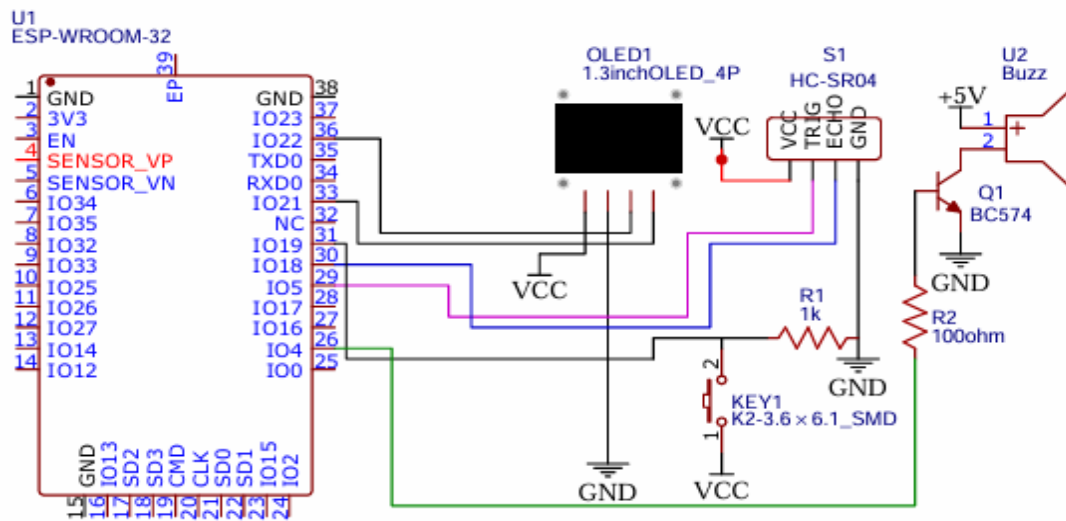


Chapter 5: Circuit Design and Implementation

This chapter explains the overall circuit layout, wiring connections, and hardware integration required for the **Ultragaug** system to function properly.

5.1 Circuit Diagram

The complete circuit is built on a **transparent 830-point solderless breadboard**. The key components—ESP32, ultrasonic sensor, OLED display, buzzer, and button—are all wired based on their pin requirements and voltage levels.



5.2 Pin Configuration

Component	ESP32 Pin	Function
HC-SR04 Trig	GPIO 5	Output trigger for pulse
HC-SR04 Echo	GPIO 18	Input to read echo time
Buzzer (via BC547)	GPIO 4	Alert buzzer control
OLED Display (I2C)	GPIO 21 (SDA) GPIO 22 (SCL)	Communication with OLED display
Button Input	GPIO 19	Unit toggle button input

Component	ESP32 Pin	Function
Power Supply	3.3V & GND	Power for sensor, display, button, etc.

5.3 Hardware Connections

1. Ultrasonic Sensor (HC-SR04):

- **VCC** → 3.3V from MB102 module
- **GND** → Common GND
- **Trig** → GPIO 5
- **Echo** → GPIO 18

2. OLED Display (1.3", SH1106, I2C):

- **VCC** → 3.3V
- **GND** → GND
- **SDA** → GPIO 21
- **SCL** → GPIO 22

3. Button:

- **One pin** → GPIO 19
- **Other pin** → GND
- **Debounce handled** via custom software class (Button.h).

4. Buzzer with Transistor Driver:

- **Buzzer VCC** → Connected to collector of BC547
- **Buzzer GND** → GND
- **Base of BC547** → GPIO 4 via 100Ω resistor
- **Emitter of BC547** → GND

5. Power Supply:

- **MB102 Breadboard Power Module** provides:
 - 3.3V → ESP32 and OLED
 - 5V → Ultrasonic sensor
 - GND → Common ground for all modules

Chapter 6: Software and Coding

This chapter outlines the tools, structure, and logic used in developing the Ultragaugue system, along with key software design strategies and challenges encountered.

6.1 Tools and IDE Used

- **PlatformIO** extension in **Visual Studio Code (VS Code)**
- **Framework:** Arduino
- **Board Configuration:** esp32dev (ESP32 WROOM32 DevKit V1)
- **Platformio.ini Configuration**

6.2 Code Structure

The project is modular and organized into key sections:

- **Main Logic (main.cpp):** Handles setup, looping, and all control logic
- **Header Files:**
 - **car_frames.h:** Stores bitmap frames for car animation (stored in PROGMEM)
 - **Button.h:** Custom debounce class for button input
 - **u8g2_oledPrint.h:** Wrapper for consistent font rendering on OLED

6.3 Animation Implementation

- Uses **U8G2** graphics library
- Download the gif file and extract their frames.
- Using image2cpp to get the HEX code, of all images, then use it array.
- Car animation is a sequence of 27 bitmap frames shown using `display.drawXBMP()`
- Using function draw each frame between the gap of 150ms.
- **Code Snippet:**

```

• void carAnimation(unsigned long duration_ms)
• {
•     unsigned long start = millis();
•     while (millis() - start < duration_ms)
•     {
•         for (int i = 0; i < 27; i++)
•         {
•             display.clearBuffer();
•             display.drawXBMP(4, 4, 120, 56, frames[i]);
•             display.sendBuffer();
•             delay(50);
•         }
•     }
• }

```


6.4 Distance Measurement and Display Logic

- Distance is measured using **HC-SR04 sensor**
- Averaging over 5 samples for stable reading
- Distance is displayed on the OLED in four modes:
 - cm
 - inch
 - feet
 - meter
 → Button toggles between modes using a cyclic switch
- **Trigger Pulse Generation**
 The ESP32 microcontroller sends a **10-microsecond HIGH pulse** to the Trigger pin of the HC-SR04 ultrasonic sensor to initiate distance measurement.
- **Ultrasonic Wave Emission**
 Upon receiving the trigger, the sensor emits an ultrasonic sound wave at a frequency of **40 kHz**.
- **Echo Reception**
 The ultrasonic wave travels through the air, and if it hits an object, it gets reflected back. The sensor's **Echo pin goes HIGH** for the duration the wave takes to return.
- **Time Duration Measurement**
 The ESP32 measures the time (in microseconds) for which the Echo pin remains HIGH. This duration represents the **round-trip time** of the ultrasonic pulse.
- **Distance Calculation**
 Using the known speed of sound in air (**0.034 cm/μs**), the distance is calculated with the formula:

$$\text{Distance (in cm)} = (\text{Duration} \times 0.034) / 2$$
 The division by 2 accounts for the to-and-fro travel of the ultrasonic pulse.
- **Real-time Loop Execution**
 The entire process is executed continuously in a loop, allowing the system to **monitor distance in real-time** and display the values on the OLED screen.
- **Collision Alert Logic**
 If the calculated distance is **less than 4 cm**, the ESP32 activates a buzzer to alert the user of a potential **crash or obstacle**.

6.5 Challenges Faced in Coding

- **Voltage mismatch** between 5V Echo pin and 3.3V ESP32 (risk of pin damage, handled via caution and optionally a voltage divider)
- **Button debounce** issues resolved via a custom library Button.h
- **OLED flickering** solved by using clearBuffer() and sendBuffer() for proper frame refresh
- **Frame memory limits** while storing animation bitmaps, managed using PROGMEM for flash memory usage

Chapter 7: Working and Output

7.1 Boot and Start-up Process

Upon powering the system, the device initiates a smooth 3-stage animated boot process on the OLED screen:

1. **Welcome Screen** – Shows animated car GIF frames to catch user attention.
2. **Project Screen** – Displays the name “*Ultragaugue*” along with the slogan “*Get Distance Digitally*”.
3. **Author Screen** – Shows “*Powered by Abhishek*”, branding the project.

This entire sequence creates a polished introduction for the user, enhancing visual appeal and branding.


7.2 Real-time Monitoring

Once the boot animation completes, the system continuously measures distance using the **HC-SR04 ultrasonic sensor** and updates the values on the OLED display in real-time.

- **Measurement Units:** The system supports cycling through multiple units using a physical push button:
 - Centimeters (cm)
 - Inches (inch)
 - Feet (ft)
 - Meters (m)
- **OLED Output Includes:**
 - A heading: “*Distance Measured*”
 - The measured value with unit (formatted neatly)
 - Warning text if object is too close (under 4cm or equivalent)

7.3 Safety Beep (Obstacle Alert)

When the measured distance goes **below 4 cm** (or corresponding unit thresholds), a **buzzer alert** is triggered, accompanied by the OLED displaying:

 “*Too Close!*”

This provides real-time safety feedback in congested or reverse parking scenarios.

7.4 Sample Outputs (Screens)

OLED Screenshots:

- Car animation frames during boot.



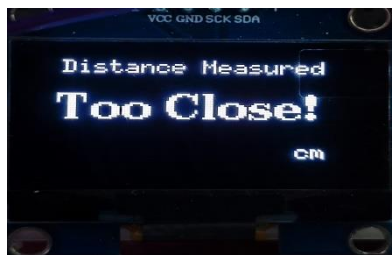
- "Welcome" screen with smooth transition.



- Distance display (in multiple units).



- "Too Close!" warning with buzzer sound.



Chapter 8: Conclusion and Future Scope

8.1 Summary of the Project

The **Ultragaugue** project demonstrates a compact and intelligent distance measurement system that provides **real-time proximity monitoring** with both **visual and auditory alerts**. Designed using the **ESP32 microcontroller**, **HC-SR04 ultrasonic sensor**, and **OLED display**, it offers:

- Animated and user-friendly interface
- Multiple distance unit options (cm, inch, feet, meter)
- Proximity-based alert mechanism with buzzer
- Portable and low-cost design

The project's boot animations and OLED visuals enhance the UX (user experience), while its application targets **smart car parking**, **congested manoeuvring**, and **obstacle prevention** in tight spaces.

8.2 Limitations

While functional and visually appealing, the current version of Ultragaugue has certain limitations:

- **Power Dependency:** Currently depends on USB or external power; no internal battery backup.
- **Short Range Only:** Effectiveness is limited to short-range measurements (within ~400 cm).
- **No Data Logging:** It doesn't store or analyse data over time.
- **Single Sensor:** Only front-facing measurement is supported.

8.3 Future Enhancements

1. **Battery Integration:** Add rechargeable battery support to make the device portable and independent of external power.
2. **Dual/Multiple Sensor Support:** Attach more sensors for side/rear monitoring for complete obstacle detection.
3. **Bluetooth/Wi-Fi Module:** Send data to a mobile app or cloud for monitoring and analysis.
4. **Mobile App Integration:** Create a simple app interface to view distance values remotely or receive alerts.
5. **Better Enclosure:** Design a compact 3D-printed casing for durability and aesthetics.
6. **Auto Calibration:** Include software-based calibration based on ambient conditions.

Chapter 9: References

9.1 Web Links

- ESP32 Documentation – Espressif
- HC-SR04 Ultrasonic Sensor Guide – Random Nerd Tutorials
- [U8g2 OLED Library Wiki – GitHub](#)
- PlatformIO Documentation
- Arduino Reference

9.2 Libraries Used

Library Name	Purpose	Installation in platformio.ini
U8g2lib	OLED Display Driver (SH1106)	U8g2@^2.33.15
Wire	I2C Communication	Built-in
Button.h (custom)	Button Handling Logic	Local header
u8g2_oledPrint.h	Text printing helper functions	Custom (included in project)

9.3 Tutorials/Communities Referred

- Arduino Forum and Stack Overflow for logic debugging and library suggestions.
- PlatformIO community for build and environment configuration guidance.
- YouTube tutorials for frame-by-frame animation on OLED.

Appendix

A.1 Complete Source Code

- You will find on my GitHub: <https://github.com/AbhiVish6386>
- Full code is available in the `src/main.cpp` file.
- Custom header files like `car_frames.h`, `Button.h`, and `u8g2_oledPrint.h` are in the `include/` directory.
- The animation byte arrays (`frames[]`) are stored using `PROGMEM` for memory optimization.

Folder Structure :

 ultragauge

```
├─ .pio
├─ .vscode
├─ include
├─ lib
├─ src
├─   └─ main.cpp
├─     └─ Button.cpp
├─       └─ car_frames.cpp
├─         └─ u8g2_oledPrint.cpp
├─           └─ Button .h
├─             └─ car_frames .h
├─               └─ u8g2_oledPrint.h
├─ platformio.ini
├─ README.md
```

THANK YOU !

By: Abhishek Vishwakarma

B.tech CSE (DS+AI) 2nd Year.