



UNIVERSITY OF VOCATIONAL TECHNOLOGY

Faculty of Engineering Technology

Department of Electro-Mechanical Technology

EE402040

Internet of Things (IoT)

IoT-Based Car Parking Assistance System

Project Proposal

Group Members

Name

Reg. no

G.G.H.D.Fernando

: MEC/22/B1/12

U.M.G.G.S.D.Udakumbura

: MEC/22/B1/18

Program

: B.Tech in Mechatronics Technology

Submission Date

: 21st June 2025

Instructed by: Mr. Janith Kasun

Contents

Chapter 1: Introduction	3-4
1.1. Abstract	3
1.2 Background	3
1.3. Problem Statement	4
1.4. Motivation.....	4
1.5. Aim and Objectives	4
Chapter 2: Literature Review.....	5-8
2.1. Scope of the Literature.....	5
2.2. Existing Solutions and Their Shortcomings	5
2.3. Identified Gaps in Current Technology	6
2.4. Supporting Research.....	7
2.5. Broader Implications and Future Directions.....	7
2.6. References	8
Chapter 3: Methodology & System Design.....	22
3.1. Architecture.....	9-10
3.2. Hardware.....	11
3.3. Software.....	12-17
3.3.1. ESP32	12-13
3.3.2. Communication.....	14
3.3.3. Data (JSON).....	15
3.3.4. Android Mobile Application.....	16-17
3.4. Network	18-19
3.5. Data Flow.....	20
Chapter 4: Implementation Plan & Timeline.....	21-22
4.1. Team Roles & Responsibilities.....	21
4.2. Project Timeline Gantt Chart.....	22
Chapter 5: Expected Outcomes & Deliverables	23
Chapter 6: Implementation Plan & Timeline.....	24-25
6.1. Budget/Resources	24
6.2. Software & Tools	25
6.3. Resources Needed.....	25
Chapter 7: References.....	26

1. Introduction

1.1. Abstract

This project introduces an IoT-based car parking assistance system that enhances driver safety and convenience using ESP32 microcontrollers, ultrasonic sensors, and a real-time feedback interface. Four ultrasonic sensors are strategically placed at the front, rear, left, and right sides of the vehicle to continuously monitor the distance to nearby obstacles. The ESP32 processes this data and transmits it wirelessly to a mobile application, which visually presents a top-down view of the car, divided into four interactive zones (front, rear, left, and right), reflecting real-time proximity data, as conceptualized in the system diagram. Each zone independently changes color (green, yellow, or red) based on the proximity of detected objects, providing intuitive and immediate visual cues to the driver. Complementing this, a buzzer is integrated into the system to deliver progressive audio alerts: it emits low-frequency beeps when obstacles are at a warning distance and switches to high-frequency beeps when an object is dangerously close. This combination of clear visual and audio feedback allows drivers to quickly identify the direction and severity of potential hazards, significantly reducing the risk of collisions and vehicle damage during parking maneuvers. The system is designed to be affordable, reliable, and easily retrofittable, making advanced parking assistance accessible for a wide range of vehicles and drivers.

1.2. Background

Parking assistance technology has evolved significantly from basic rear-mounted sensors introduced in the 1990s to sophisticated multi-sensor arrays integrated with vehicle control systems today. Early systems relied on simple ultrasonic technology providing audible alerts, while modern systems incorporate visual feedback, camera integration, and even semi-autonomous parking capabilities. This evolution has been driven by increasing vehicle sizes, growing urban congestion, and consumer demand for enhanced safety features. The global automotive parking assistance system market has experienced substantial growth, valued at approximately \$11.5 billion in 2023 and projected to reach \$22.3 billion by 2030. Current technologies predominantly utilize ultrasonic sensors due to their cost-effectiveness and reliability, though radar, cameras, and infrared systems are increasingly common in premium vehicle segments. This project stands out by focusing on affordability, modularity, and ease of integration for non-premium vehicles, which are often underserved by advanced safety technology.

1.3. Problem Statement

Parking in confined or crowded spaces is a significant challenge for drivers, often resulting in accidents, vehicle damage, and stress. Limited visibility around the vehicle perimeter makes accurate distance judgment difficult, especially at night, with larger vehicles, or for drivers with mobility limitations. Drivers face unnecessary risks and potential vehicle damage during parking maneuvers without reliable assistance systems.

"According to insurance data, low-speed parking accidents account for a significant portion of vehicle damage claims, especially in urban environments."

1.4. Motivation

Urbanization and increased vehicle density have made efficient, affordable, and user-friendly parking assistance systems essential. Existing solutions are either expensive, complex, or lack intuitive feedback. An IoT-based system leveraging modern microcontrollers and wireless communication can democratize access to advanced parking assistance, making roads safer and reducing repair costs for all drivers.

1.5. Aim & Objectives

Aim:

To design and implement a comprehensive, IoT-based car parking assistance system using ESP32 microcontrollers and ultrasonic sensors, with real-time visual feedback on a mobile dashboard.

Objectives:

- **Sensor Integration:** Connect four ultrasonic sensors (front, rear, left, right) with ESP32 to enable 360° obstacle detection with ± 2 cm accuracy up to 2 meters.
- **User Interface:** Build a Python-based web dashboard (or Android app) displaying real-time top-down proximity view, with adaptive color-coded zones.
- **Communication Protocols:** Use MQTT/HTTP for reliable wireless data transmission, structured using JSON.
- **System Completion:** Deliver a functional hardware prototype and software interface within the defined project schedule.

2. Literature Review

2.1. Scope of the Literature

As cities grow and vehicle density increases, safe and convenient parking has become a major concern for drivers. Navigating narrow streets, tight parking lots, and unfamiliar surroundings requires precise spatial awareness. Traditional methods, such as using side and rear-view mirrors, are often inadequate due to their limited fields of view and inability to detect low-lying or side obstacles. Even experienced drivers may struggle, leading to costly damages and accidents.

Modern solutions like rear-view cameras and proximity sensors have made significant contributions, but they remain limited by high costs, limited compatibility with older vehicles, and a lack of scalability. These constraints highlight the pressing need for a low-cost, modular, and easily deployable system.

This review examines existing technologies, identifies their limitations, and positions our proposed IoT-based obstacle detection and visual feedback system as a comprehensive alternative. Our system utilizes ESP32 microcontrollers connected to ultrasonic sensors, wirelessly communicating with a mobile application that offers intuitive, real-time feedback.

2.2. Existing Solutions and Their Shortcomings

A. Traditional Mirror-Based Parking

Using side mirrors and rear-view mirrors is the most basic method of assisting with parking. While inexpensive and universally available, this method depends entirely on the driver's judgment. It lacks coverage for blind spots and is ineffective at detecting small or low-lying obstacles. According to studies, a significant portion of low-speed parking accidents occur due to these limitations.

B. Rear-View Camera Systems

Rear-view cameras provide live video feedback from the rear of the vehicle and are standard in many modern cars. However, they are prone to environmental interference such as mud, rain, or glare can obscure the view. Additionally, their installation and calibration can be complex and expensive. These systems focus mainly on the rear and lack 360° detection.

2.3. Identified Gaps in Current Technology

Despite many advancements, current solutions have gaps:

- **High Cost and Limited Access:** Many systems are too expensive for average users or retrofitting into older cars.
- **Directional Limitation:** Most systems only cover the rear, ignoring side and front zones.
- **Poor User Interface Design:** Beeping alarms are difficult to interpret in noisy environments and may induce stress.
- **Lack of Modularity:** Proprietary systems are not open-source or customizable for different use cases.

Our system directly addresses these gaps with:

- **ESP32 Microcontroller Network:** Multiple ESP32 units, each with an ultrasonic sensor, form a distributed system that communicates via Wi-Fi or Bluetooth.
- **Real-Time Visual Feedback:** A mobile dashboard app provides intuitive color-coded distance indicators.
- **Modular and Expandable Design:** Additional zones can be added without altering the core architecture.
- **Low-Cost Implementation:** The use of affordable, widely available components ensures broad accessibility.

This project thus serves as a bridge between costly commercial systems and basic manual methods, providing a smart, affordable, and scalable solution.

2.4. Supporting Research

The foundation of our system is supported by existing research:

- **Sensor Accuracy:** Jain and Singh [1] demonstrated that ultrasonic readings can be significantly improved through software-level noise filtering. This supports our approach of preprocessing sensor data locally on each ESP32.
- **Dashboard Design:** Nagy and Sándor [2] emphasized the importance of reducing cognitive load in dashboard designs. Inspired by their work, our app uses simple, clear indicators that adapt based on distance severity.
- **Structured Testing:** Buchholz et al. [3] introduced SHPbench, a simulation-based validation framework. While our system does not require such complexity, we incorporate structured, scenario-based testing to ensure real-world reliability.

These works validate the feasibility, reliability, and user-centered approach of our system.

2.5. Broader Implications and Future Directions

Our solution not only serves personal vehicles but has potential for broader applications:

- Logistics and Warehouse Vehicles: Enhancing safety in tight warehouse environments
- Public Transport: Assisting in maneuvering large vehicles in congested urban areas
- Agricultural Equipment: Helping in obstacle detection in remote or uneven terrain

Future development may include:

- Integration with GPS for spatial awareness
- Voice feedback options for visually impaired users
- Web-based dashboard for remote monitoring

2.6. References

[1] S. Jain and K. Singh, "Signal Noise Reduction Algorithm for Object Localization and Tracking Using a 2-Dimensional Ultrasonic Sensor Array," 2024 8th Int. Conf. on Computational System and Information Technology for Sustainable Solutions (CSITSS), Bengaluru, India, 2024, pp. 1-6. doi: 10.1109/CSITSS64042.2024.10816775.

Available: <https://ieeexplore.ieee.org/document/10816775>

[2] V. Nagy and Á.P. Sándor, "Dynamic Vehicle Dashboard Design for Reduced Driver Distraction," in Machine and Industrial Design in Mechanical Engineering, vol. 174, Springer, Cham, 2025. doi: 10.1007/978-3-031-80512-7_63.

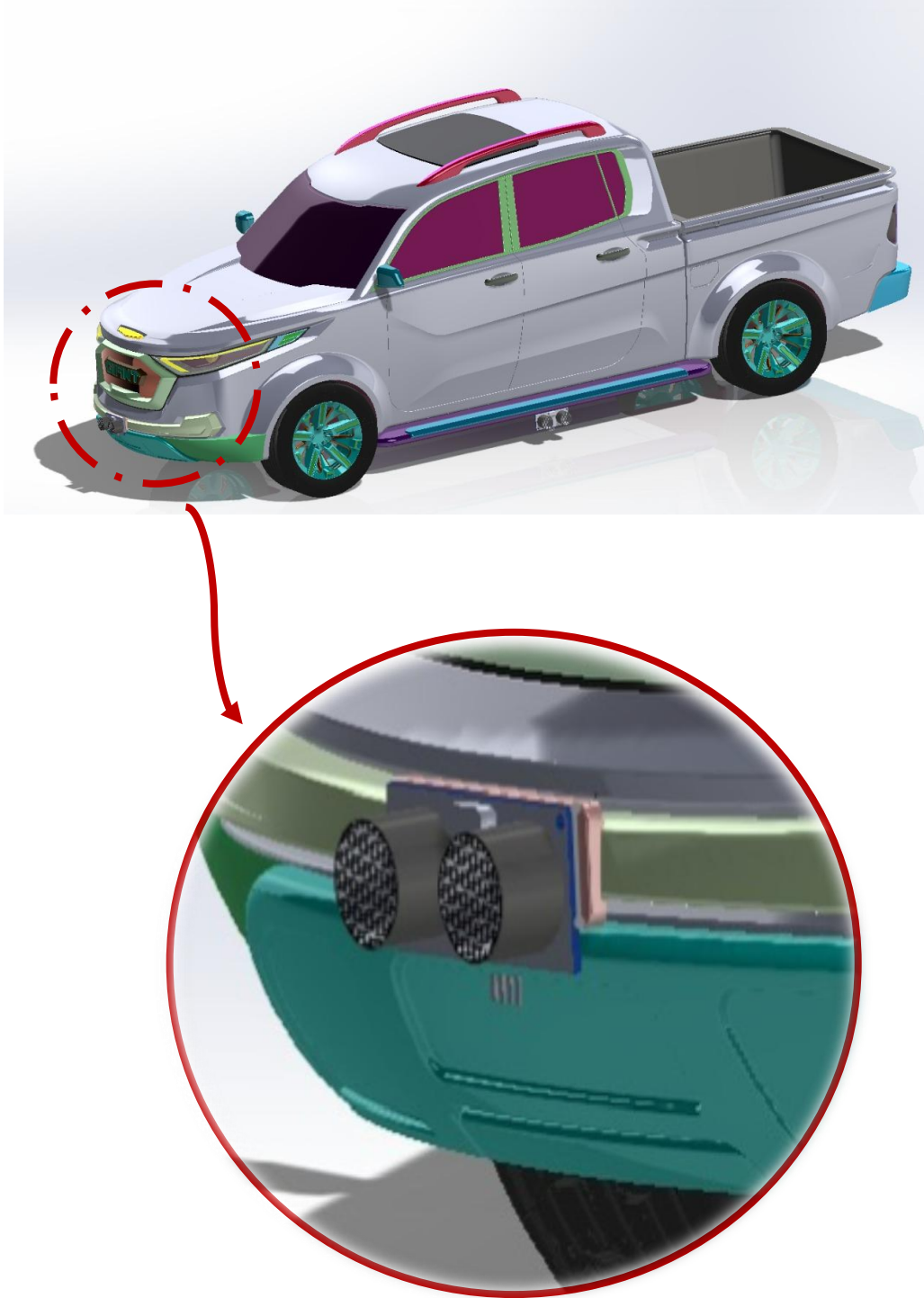
Available: https://link.springer.com/chapter/10.1007/978-3-031-80512-7_63

[3] C. Buchholz, T. Vorsatz, S. Kind, and R. Stark, "SHPbench – A Smart Hybrid Prototyping Based Environment for Early Testing, Verification and (user based) Validation of Advanced Driver Assistant Systems of Cars," Procedia CIRP, vol. 60, pp. 139–144, 2017. doi: 10.1016/j.procir.2017.02.025.

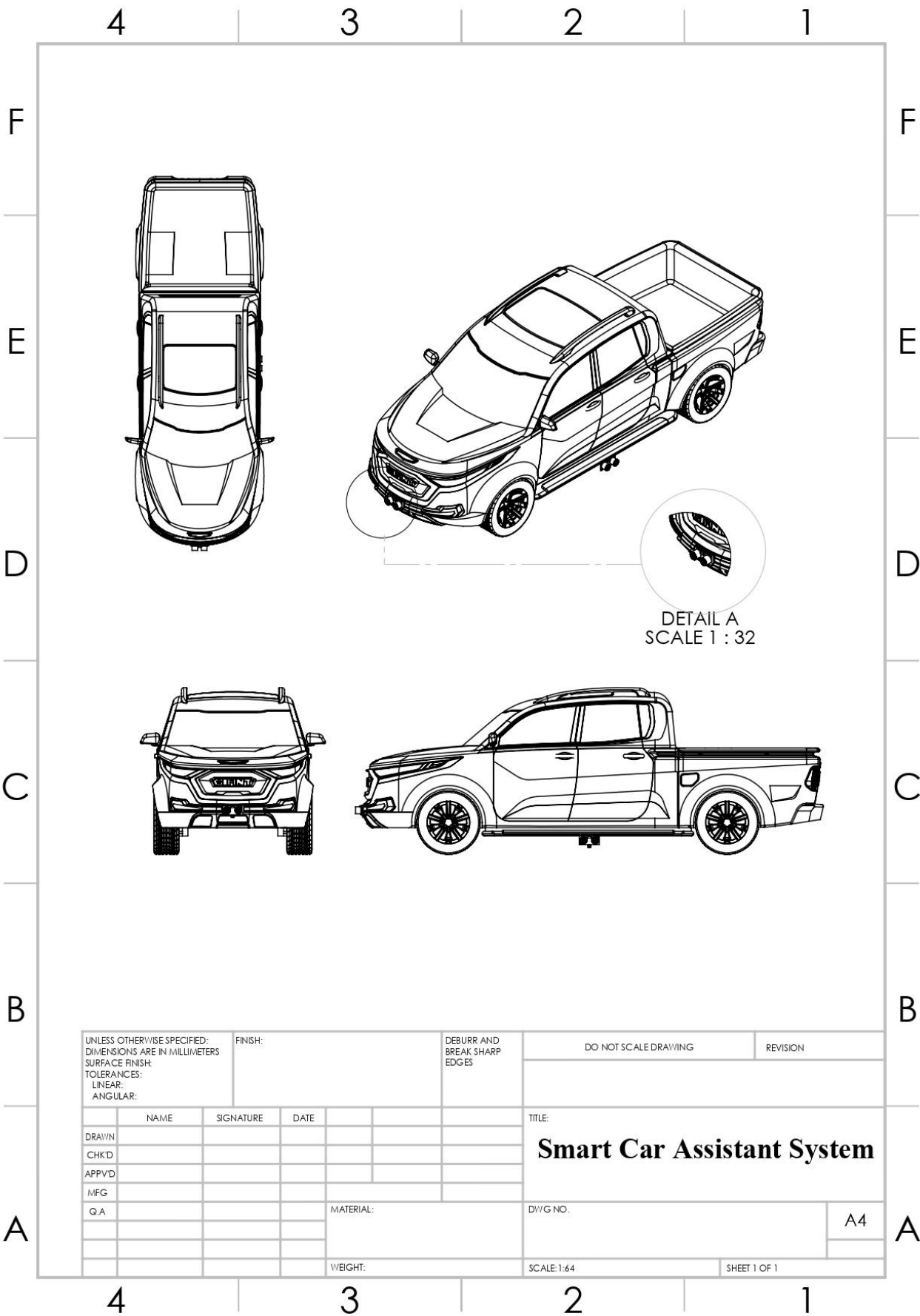
Available: <https://www.sciencedirect.com/science/article/pii/S221282711730121X>

3. Methodology & System Design






3.1. Architecture



Solidworks Design

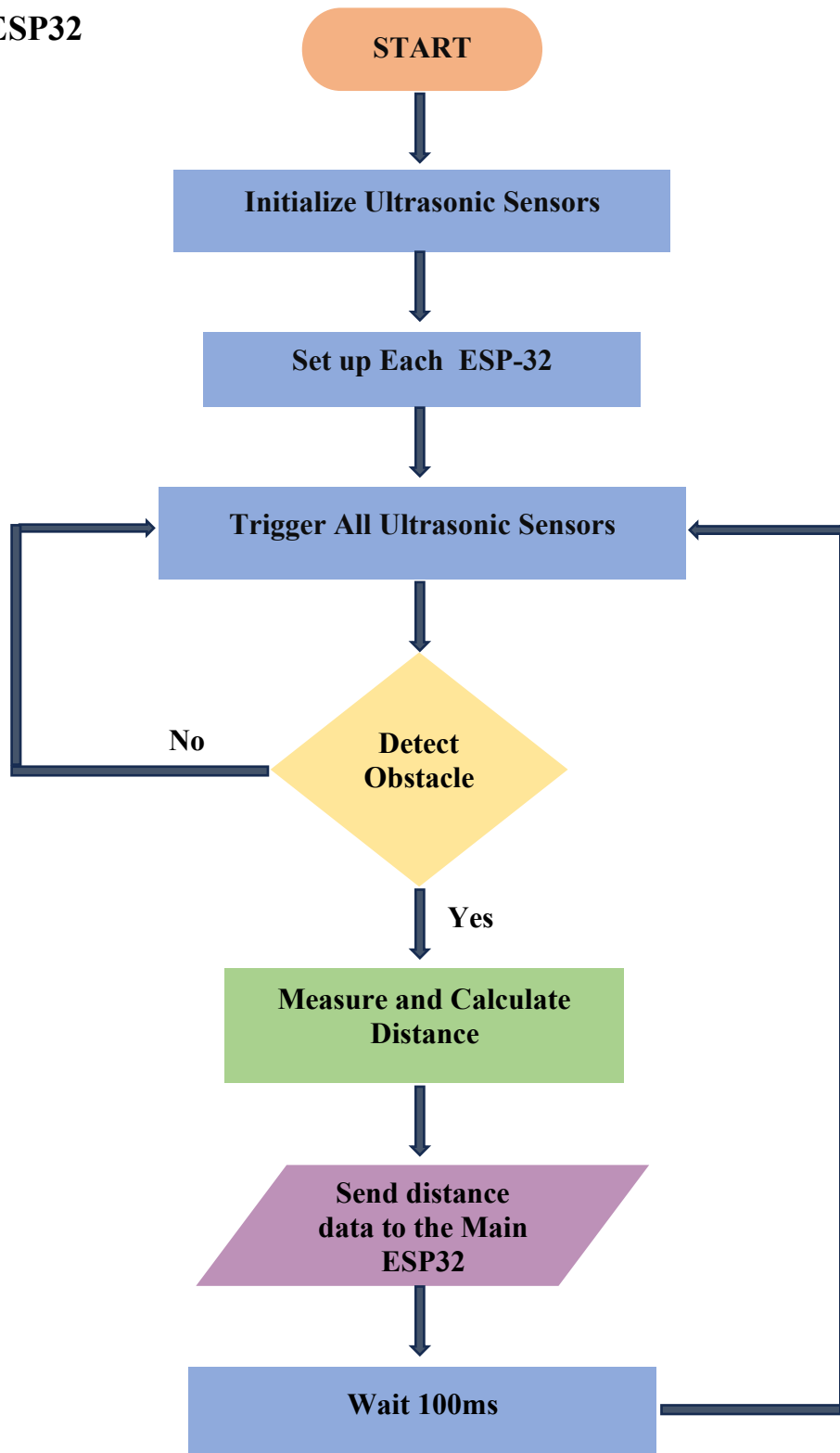


3.2. Hardware

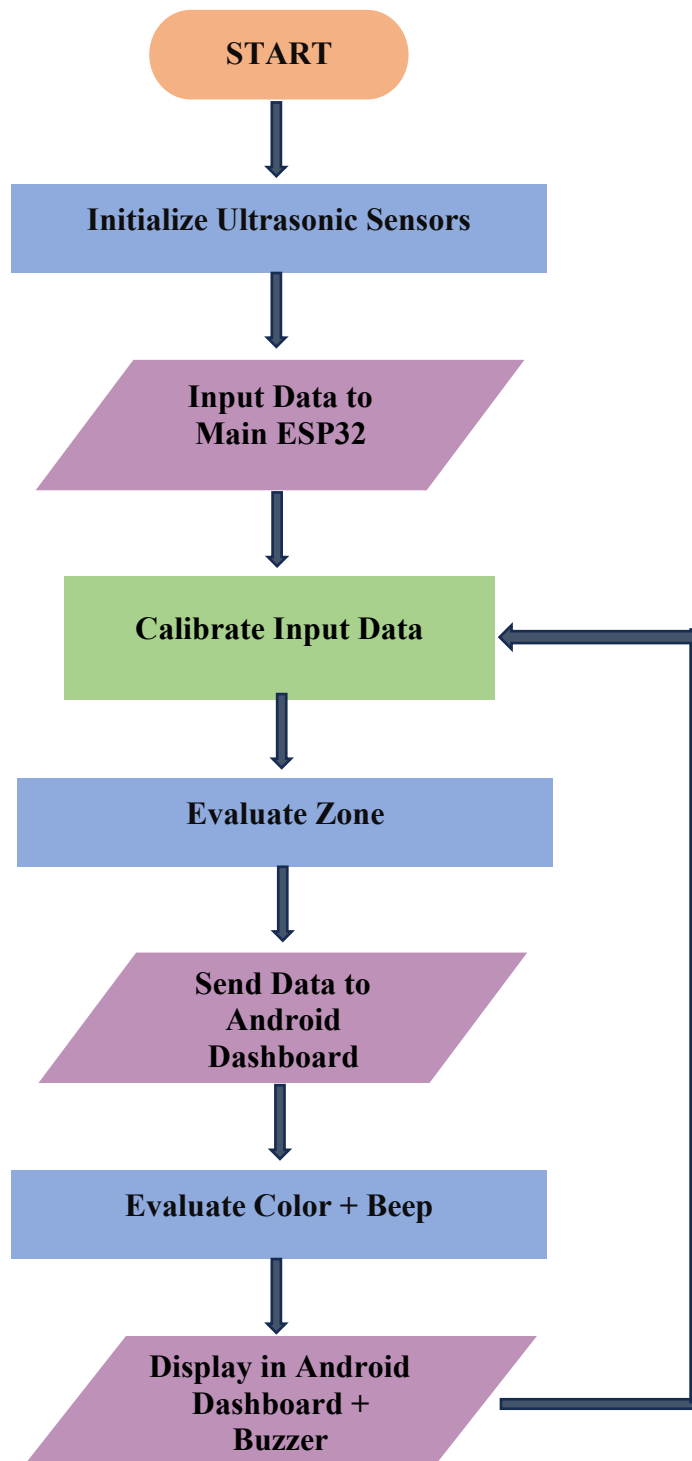
Component	Qty.	Justify	Picture
ESP32	5	<ul style="list-style-type: none"> Dual-core MCU with built-in Wi-Fi/Bluetooth Controls sensors and handles wireless communication 	
Ultrasonic sensor (HC-SR04 Sensor)	4	<ul style="list-style-type: none"> Measures distance using sound waves Range: 2 cm to 400 cm 	
Buzzer	1	<ul style="list-style-type: none"> Emits beeping sounds for danger alerts Beeping frequency increases as the obstacle nears 	
Android TV	1	<ul style="list-style-type: none"> Mounted on the dashboard to act as a live interface Shows top-down view with visual distance alerts 	
5V battery	5	<ul style="list-style-type: none"> Power for each ESP32 (e.g., power bank or USB) 	

3.3. Software

3.3.1. ESP32



This flowchart shows how an ESP32 sensor node works in a parking assistant system. It initializes sensors and sets up ESP-NOW communication, then continuously measures distance and sends the data with an ID to the central ESP32 for real-time monitoring.



This flowchart outlines the main ESP32 controller's process. It initializes Wi-Fi, ESP-NOW, and I/O pins, then waits for sensor data via a registered callback. When data arrives, it updates the relevant sensor reading, checks if the distance triggers a warning or danger alert, activates the buzzer if needed, and sends a JSON update to the Android TV. This cycle repeats as new data is received.

3.3.2. Communication

Chosen Communication Method: HTTP

In this smart car parking assistant system, HTTP was selected as the communication protocol between the main ESP32 controller and the Android TV dashboard. This decision was based on HTTP's flexibility, simplicity, and compatibility with modern display devices such as Android TV.

The internal communication between sensor ESP32 nodes and the main ESP32 controller is managed using ESP-NOW, a lightweight, connectionless protocol developed by Espressif. ESP-NOW is ideal for low-latency, device-to-device communication and operates without requiring a router or internet access, making it well-suited for local, fast, and efficient data transmission from sensor nodes.

Why was HTTP chosen?

- **No Internet Required:** The ESP32 can act as a local web server or Access Point (AP), allowing HTTP-based communication even in offline scenarios.
- **Cross-Platform Compatibility:** Android TV supports modern web browsers that can access local IP addresses, making HTTP a practical method for real-time data visualization.
- **Data Presentation:** The HTTP server on the ESP32 can serve live HTML pages, JavaScript dashboards, or raw JSON data. This enables dynamic visual displays of parking status, distance alerts, or other system metrics.
- **Ease of Development:** HTTP is a widely used protocol with simple request/response mechanisms (e.g., GET, POST), supported by many development tools and libraries, reducing development complexity.
- **Low Resource Demand:** For relatively small payloads like sensor distances or occupancy status, HTTP is light enough to perform reliably on the ESP32 without straining its memory or CPU.
- **Polling and Refreshing:** The Android TV can use techniques like AJAX or WebSockets (optional future upgrade) to fetch updated data periodically, offering near real-time updates without requiring frequent page reloads.

3.3.3. Data (JSON)

In this project, **JSON (JavaScript Object Notation)** is used as the format to send real-time sensor data from the **main ESP32** to the **Android TV interface**. JSON was chosen because it is:

- Easy to read and write
- Lightweight
- Well-supported by web browsers and JavaScript
- Ideal for structuring multi-sensor data in one compact message

What Data Is Sent?

The main ESP32 collects distance readings from all four sensor nodes (front, rear, left, right) and packages them into a single JSON object. This data is served via an HTTP endpoint (e.g., /status) and can be parsed by the Android TV dashboard.

Field	Type	Description
Front, Rear, Left, Right	Integer	Distance to obstacle from each side (in/cm)
Alert	Object	Contains the danger level of each side based on distance

Use of JSON in System

- The Android TV UI fetches this JSON periodically using JavaScript (e.g., with fetch()).
- It then updates the top-view dashboard by changing colors or zones.
- This keeps the UI synced with the latest sensor readings.

3.3.4. Android Mobile Application

Introduction

The Android mobile application serves as the primary user interface for the IoT-based obstacle detection system. It receives real-time distance data from the central ESP32 controller and provides visual and auditory feedback to the user, enabling enhanced awareness of surrounding obstacles.

Features

- Real-time display of obstacle distance data from four zones: front, rear, left, and right.
- Color-coded zone indicators (**Green: Safe, Yellow: Caution, Red: Danger**).
- Buzzer activation feedback based on distance thresholds (audible alerts for critical proximity).
- Simple and clean layout for quick interpretation.
- Auto-connect capability with the ESP32 via Bluetooth or Wi-Fi.

Framework & Tools Used

- Platform: Android (built using MIT App Inventor / Android Studio / Flutter/Blynk).
- Communication: Bluetooth Serial / Wi-Fi Socket.
- Design: XML-based layout or drag-and-drop GUI blocks.
- Data Format: Plain text or delimited string

UI/UX Design

The UI follows user-friendly and distraction-free design principles:

- Clear segmentation of the four sensor zones.
- Dynamic zone color updates based on received data.
- Bold fonts and intuitive color schemes.
- Visual feedback synchronized with buzzer alerts for better responsiveness.

Communication with ESP32

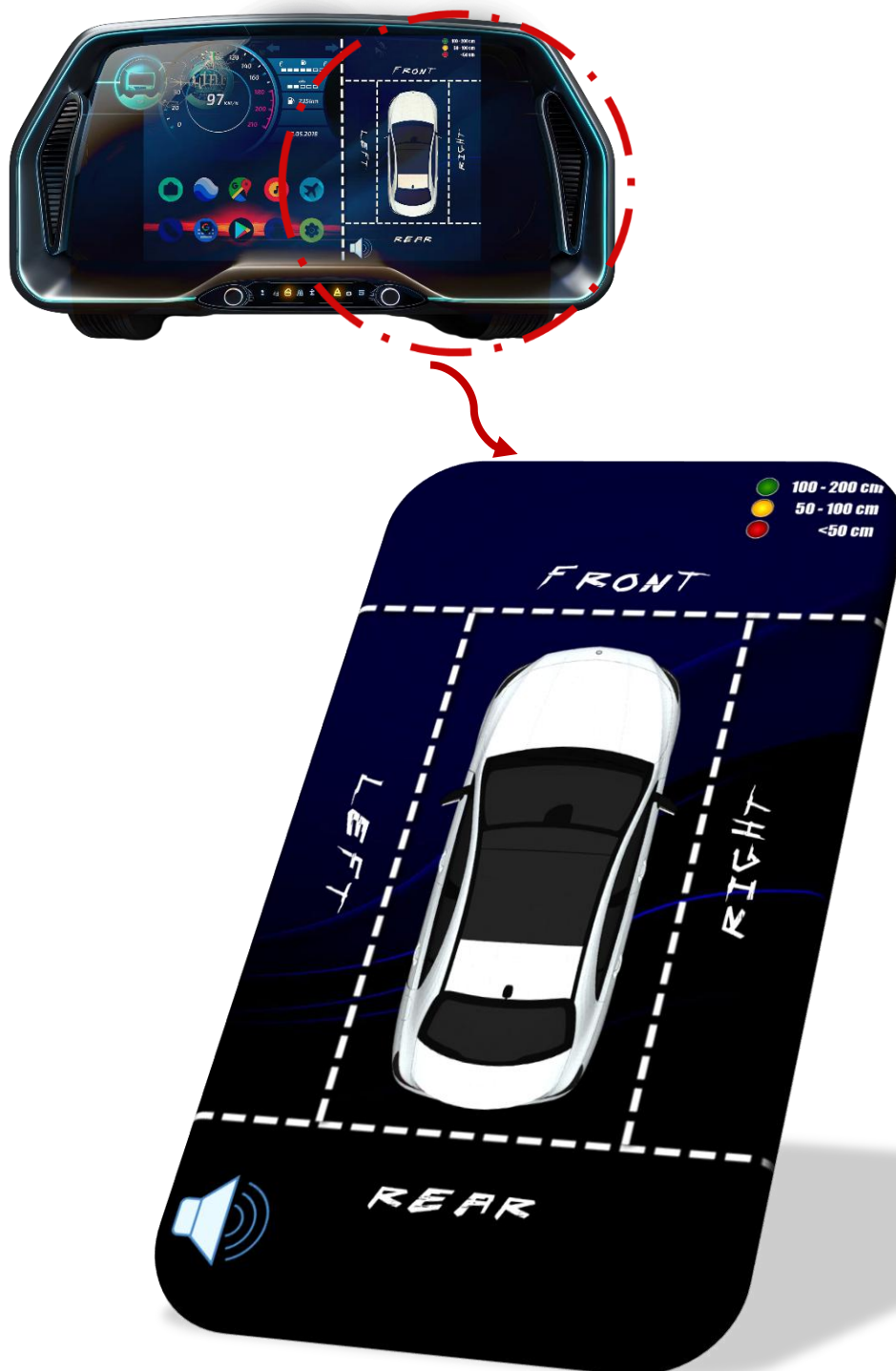
The app communicates with the central ESP32 using either:

- Bluetooth Serial: Direct pairing and serial text parsing.
- Wi-Fi: UDP or TCP communication for longer-range flexibility.

Each data packet from ESP32 contains all zone distances. The app parses this and updates the dashboard UI accordingly, while triggering buzzer alerts if any zone exceeds the danger threshold.

The app is designed to be responsive, efficient, and easily deployable on any Android device, making it ideal for real-time vehicle obstacle monitoring.

User Interface



3.4. Network

Network Architecture: Communication Layer Overview

The overall system architecture is designed as a three-tier communication model to ensure modularity, expandability, and reliable data transmission from sensors to the user interface.

Layer 1: Sensor Layer (Ultrasonic Sensors → Local ESP32 Nodes)

Each ultrasonic sensor module is directly connected to a dedicated ESP32 development board. These four ESP32 nodes function independently and are responsible for:

- Continuously measuring object distances using ultrasonic pulses.
- Assigning a unique sensor ID (e.g., front, rear, left, right).
- Packaging the measured distance with its ID.
- Preparing the data for transmission to the Main Controller via Bluetooth.

Layer 2: Local Wireless Sensor Network (ESP32 Nodes → Main ESP32 via ESP-NOW)

All four ESP32 sensor nodes communicate wirelessly with the central ESP32 controller (Main ESP32) using ESP-NOW, a proprietary connectionless wireless communication protocol developed by Espressif. This layer acts as the internal communication backbone and features:

- Lightweight, low-latency, and connectionless data exchange over the 2.4 GHz Wi-Fi band without the need for a router or access point.
- Supports simultaneous parallel reception from multiple sensor nodes via registered callback functions.
- Real-time parsing, processing, and synchronization of incoming sensor data packets for accurate and timely obstacle detection.

Layer 3: Interface Layer (Main ESP32 → Android TV Dashboard via Wi-Fi)

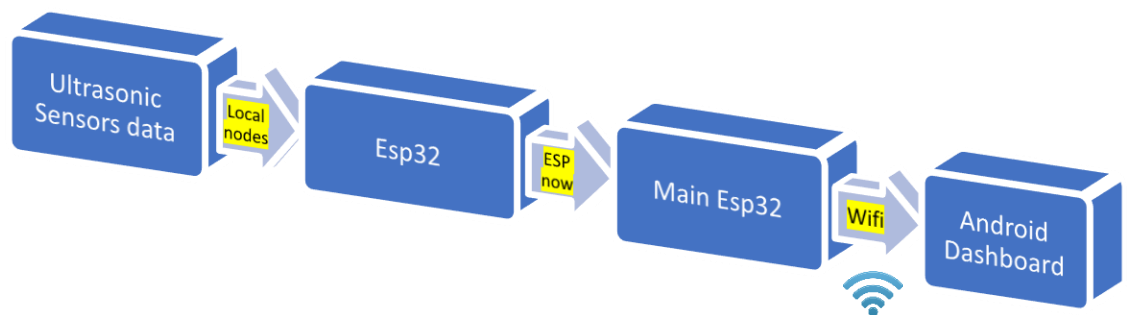
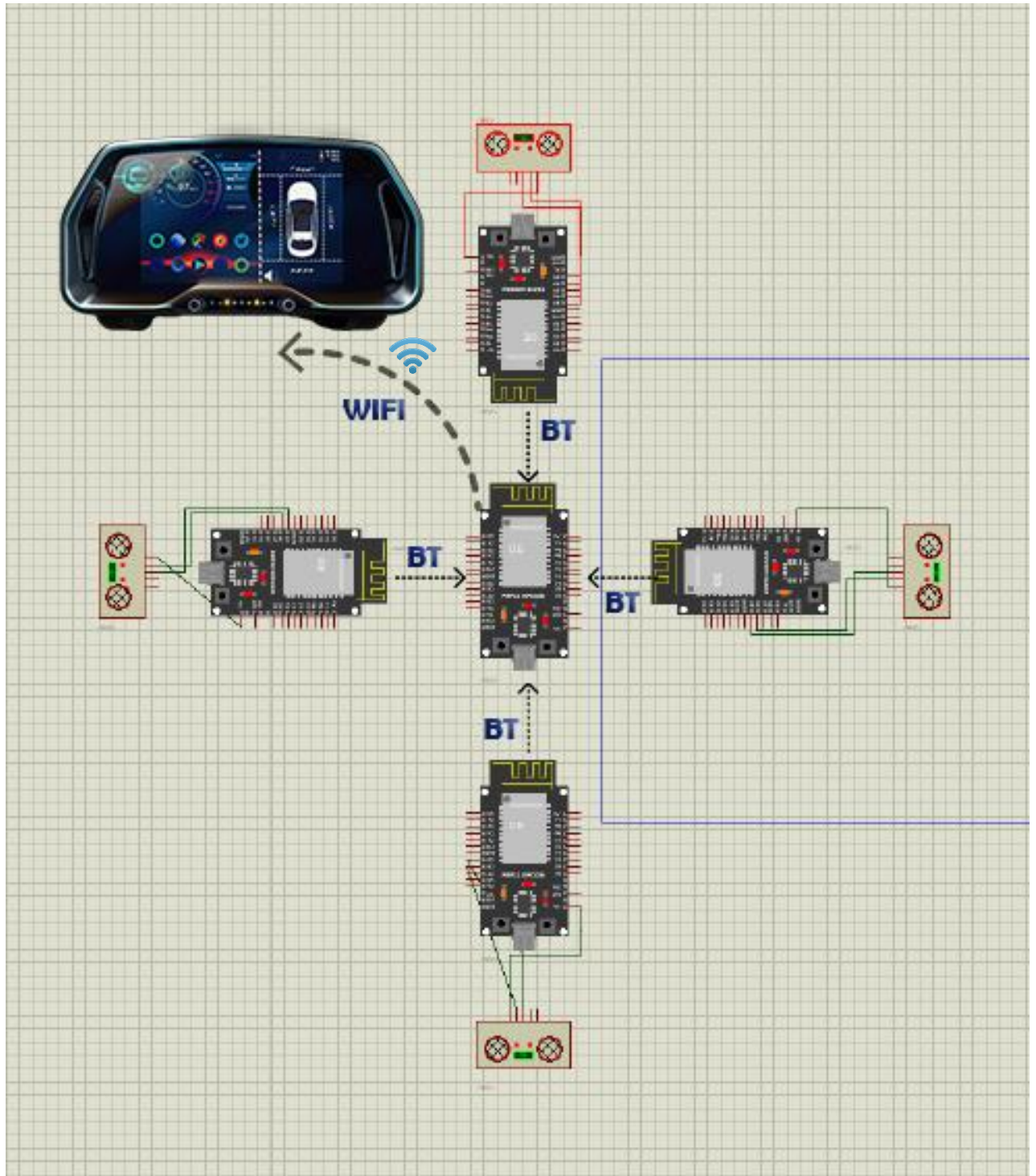
Once the Main ESP32 aggregates all the sensor data and performs necessary processing (e.g., distance interpretation, zone classification), it transmits the compiled obstacle status to the visual interface:

- Data is sent wirelessly via Wi-Fi to an Android-based dashboard running on a mobile device or smart TV.
- The dashboard displays proximity status using color-coded visual indicators for each direction (front, rear, left, right).
- The Wi-Fi channel ensures longer-range and higher-bandwidth communication suitable for GUI updates.

Summary:

This 3-layer architecture enables distributed sensing, centralized decision-making, and intuitive user feedback over efficient wireless links. The modular design also allows easy expansion by adding more sensors or interface options in future versions.

3.5. Data Flow



4. Implementation Plan & Timeline

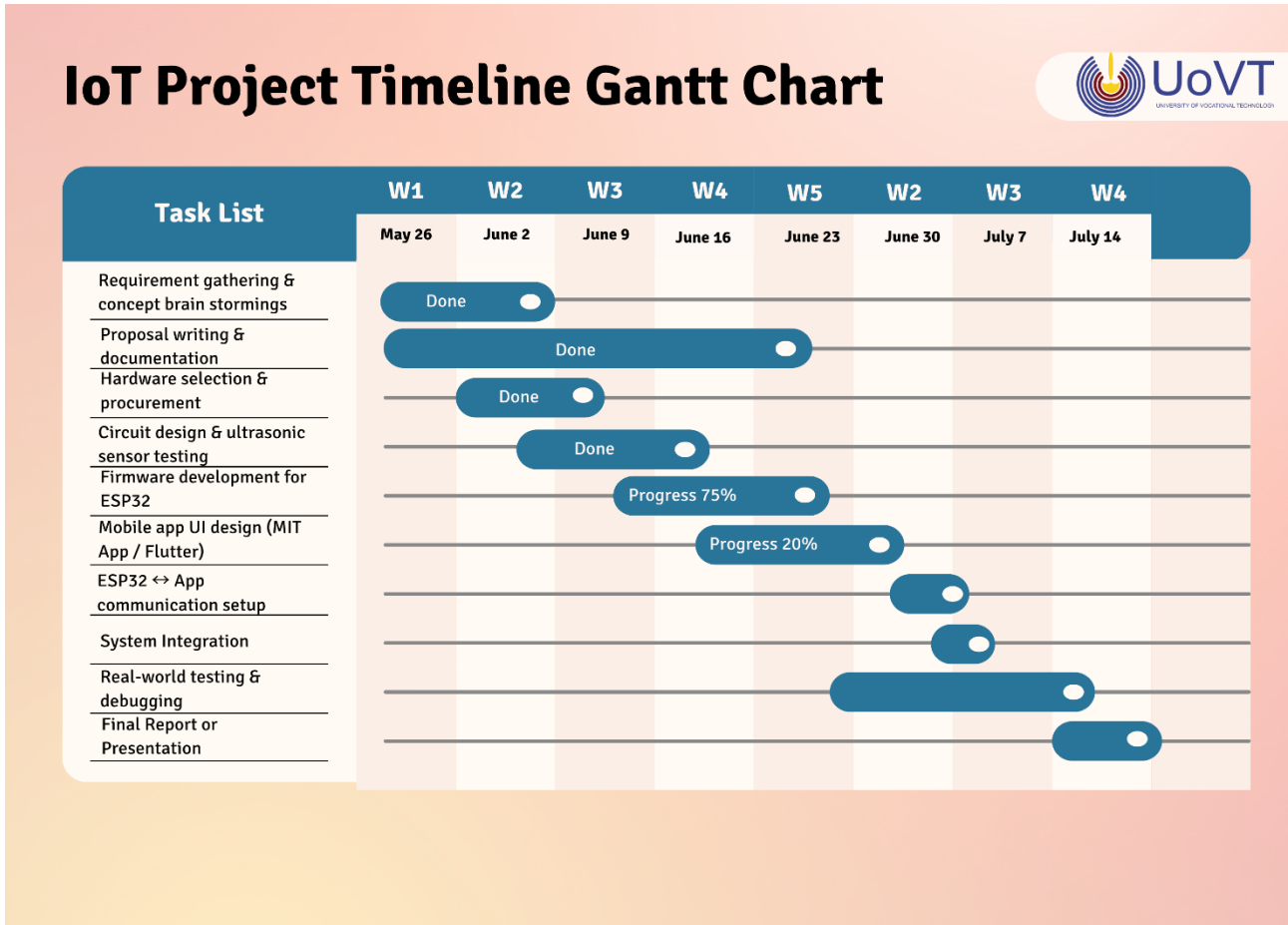
Project Members:

- U.M.G.G.S.D. Udakumbura (Reg. No: MEC/22/B1/18)
- G.G.H.D. Fernando (Reg. No: MEC/22/B1/12)

4.1. Team Roles & Responsibilities

Tasks	Responsible Member(s)	Notes
Requirement Analysis & Initial Planning	Both	Brainstorming, technical feasibility, system sketching
Hardware Procurement & Setup	Udakumbura	Selecting sensors, ESP32, Jumper wires, Breadboards, Battery supply, Buzzer, Android Display
Circuit Design & Assembly	Fernando	Testing ESP32 + Ultrasonic wiring on breadboard
Firmware Programming (ESP32)	Both	Coding with Arduino IDE, handling sensor readings
Mobile App Design & Dashboard UI	Both	Visualizing feedback in real time using MIT App Inventor / Flutter
Communication Setup (ESP32 ↔ App)	Both	Testing Bluetooth/WiFi messaging between devices
System Integration	Both	Assembling sensors, ESP32s, the app, and testing together
Testing & Troubleshooting	Both	Real-world obstacle detection testing in mock setup
Documentation & Report Preparation	Udakumbura	Final proposal, mid-report, and final report formatting
Presentation Preparation	Both	Scriptwriting, editing

4.2. Project Timeline Gantt Chart



5. Expected Outcomes & Deliverables

At the end of the project, we expect to deliver the following tangible and measurable outputs:

- **Functional Prototype**

A working IoT-based obstacle detection system using multiple ESP32 microcontrollers and ultrasonic sensors. The system will be tested in a controlled environment to demonstrate real-time obstacle detection and visual feedback.

- **Mobile Dashboard Application**

A fully developed mobile application (built using MIT App Inventor or Flutter) capable of displaying real-time obstacle proximity through color-coded visuals, enabling the user to respond instantly and accurately.

- **Firmware & Source Code**

Well-commented Arduino (ESP32) source code for the ultrasonic sensors and communication logic, along with the mobile application source files. These will be version-controlled and uploaded to GitHub or a ZIP folder.

- **Testing Report**

Detailed documentation of system performance under different conditions (e.g., object distance, sensor accuracy, environmental interference) supported by charts, images, and test cases.

- **Final Report**

A structured and comprehensive report including problem background, methodology, system design, results, analysis, and conclusion.

6. Budget / Resources

Below is the estimated budget and list of required resources for implementing and testing the system.

6.1. Hardware & Components

Item	Quantity	Unit Price (Rs.)	Total Cost (Rs.)
ESP32 Microcontroller	5	1,340	6,700
HC-SR04 Ultrasonic Sensor	4	350	1,400
Breadboard + Jumper Wires	one set	800	800
8V Battery	5	500	2,500
Miscellaneous (tape, glue)	-	500	500
Mobile device for testing	1	-	personal use
Laptop for programming	2	-	personal use

Estimated Total: Rs. 11,900.00

6.2. Software & Tools

- Arduino IDE (Free)
- MIT App Inventor / Flutter SDK (Free)
- Excel / Word / PowerPoint

6.3. Resources Needed

- Stable internet connection
- Power supply for sensor testing
- Access to testing space (indoor or outdoor)
- Phone for test the software

7. References

- 1) https://en.wikipedia.org/wiki/Ultrasonic_transducer
- 2) https://flutter.dev/?utm_source=google&utm_medium=cpc&utm_campaign=brand_sem&utm_content=apac_apac&gclid=Cj0KCQjwjdTCBhCLARIsAEu8bpKcefmpYA-QFDUtjuX-9S0LweCzeJYRgQBNSbZjejd7d46lgZbGtz8aAu-EEALw_wcB

Research Papers

- 3) <https://ieeexplore.ieee.org/document/10816775>
- 4) https://link.springer.com/chapter/10.1007/978-3-031-80512-7_63
- 5) <https://www.sciencedirect.com/science/article/pii/S221282711730121X>