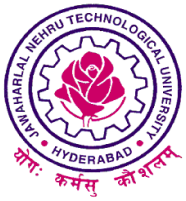
AUTONOMOUS CAR PARKING SYSTEM

A Industry Oriented Mini Project Report

Submitted to



Jawaharlal Nehru Technological University Hyderabad

In partial fulfillment of the requirements for the

award of the degree of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS & COMMUNICATION ENGINEERING

By

CHADA VARSHA (22VE1A0410)

MANGA HEMANTH (22VE1A0434)

JODUMASIDULA GANESH (22VE1A0421)

Under the Guidance of

MRS. N. CHANDANA

Asst. Professor

SREYAS
INSTITUTE OF ENGINEERING AND TECHNOLOGY
AUTONOMOUS

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

Approved by AICTE, New Delhi | Affiliated to JNTUH, Hyderabad | Accredited by NAAC "A" Grade & NBA| Hyderabad | PIN: 500068

(2022– 2026)

AUTONOMOUS CAR PARKING SYSTEM

A Industry Oriented Mini Project Report

Submitted to

Jawaharlal Nehru Technological University Hyderabad

In partial fulfillment of the requirements for the

award of the degree of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS & COMMUNICATION ENGINEERING

CHADA VARSHA (22VE1A0410)

MANGA HEMANTH (22VE1A0434)

JODUMASIDULA GANESH (22VE1A0421)

Under the Guidance of

MRS. N. CHANDANA

Asst. Professor

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

Approved by AICTE, New Delhi | Affiliated to JNTUH, Hyderabad | Accredited by NAAC "A" Grade & NBA| Hyderabad | PIN: 500068

(2022 – 2026)

SREYAS
INSTITUTE OF ENGINEERING AND TECHNOLOGY
AUTONOMOUS

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

Approved by AICTE, New Delhi | Affiliated to JNTUH, Hyderabad | Accredited by NAAC "A" Grade & NBA| Hyderabad | PIN: 500068

Certificate

This is to certify that the Industry Oriented Mini Project Report on ***"Autonomous car parking system***" submitted by **CH. Varsha, M. Hemanth, J. Ganesh** bearing Hall Ticket No's. **22VE1A0410, 22VE1A0434, 22VE1A0421** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Electronics & Communication Engineering** from Jawaharlal Nehru Technological University, Kukatpally, Hyderabad for the academic year 2024-25 is a record of bonafide work carried out by him / her under our guidance and Supervision.

Guide **Head of the Department**

Project Coordinator **Signature of the External Examiner**

SREYAS
INSTITUTE OF ENGINEERING AND TECHNOLOGY
AUTONOMOUS

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

Approved by AICTE, New Delhi | Affiliated to JNTUH, Hyderabad | Accredited by NAAC "A" Grade & NBA| Hyderabad | PIN: 500068

DECLARATION

We, **CH. Varsha, M. Hemanth, J. Ganesh**, bearing **Roll No's 22VE1A0410, 22VE1A0434, 22VE1A0421** hereby declare that the Project titled ***"Autonomous car parking system"*** done by us under the guidance of Mr. B. Sreenivasu, which is submitted in the partial fulfillment of the requirement for the award of the B.Tech degree in **Electronics & Communication Engineering** at **Sreyas Institute of Engineering & Technology** for Jawaharlal Nehru Technological University, Hyderabad is our original work.

Chada Varsha (22ve1a0410)

Manga Hemanth (22ve1a0434)

Jodumasidula Ganesh (22ve1a0421)

ACKNOWLEDGEMENT

The successful completion of any task would be incomplete without mention of the people who made it possible through their guidance and encouragement crowns all the efforts with success.

We take this opportunity to acknowledge with thanks and deep sense of gratitude to **Mrs.N.Chandana** for his constant encouragement and valuable guidance during this work.

A Special note of Thanks to **Head of the Department,** who has been a source of Continuous motivation and support in the completion of this project. He had taken time and effort to guide and correct us all through the span of this work.

We owe very much to the **Department Faculty, Principal and the Management** who made our term at **Sreyas Institute of Engineering and Technology** a Steppingstone for my career. We treasure every moment we had spent in the college.

Last but not least, our heartiest gratitude to our parents and friends for their continuous encouragement and blessings. Without their support this work would not have been possible.

Chada Varsha(22VE1A0410)

Manga Hemanth (22VE1A0434)

Jodumasidula Ganesh (22VE1A0421)

ABSTRACT

The Autonomous Car Parking System is a smart vehicle navigation solution designed to automate the parking process without human intervention. This project utilizes the ESP32 microcontroller as the central processing unit to coordinate sensor inputs and motor controls. The system integrates L298N motor driver to manage the motion of L gear motors, enabling precise movement and steering during parking operations. Power is supplied through a rechargeable battery pack, regulated by a buck controller to ensure stable voltage, while a charging module and adapter facilitate convenient recharging.

Connectivity among the components is achieved using jumper wires, ensuring modularity and easy maintenance. The ESP32's built-in Wi-Fi and Bluetooth capabilities allow for remote monitoring and control, potentially through a mobile application. This project demonstrates a cost-effective, scalable approach to address the growing need for intelligent parking solutions in urban environments, aiming to reduce human error, save time, and improve overall efficiency in vehicle management systems.

1. Keywords:
   Autonomous Navigation,ESP32 Microcontroller,Smart Parking System,L298N Motor Driver,Ultrasonic Sensor Integration,Bluetooth Vehicle Control

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

1. Introduction

1.1 Background:

In many cities, parking-related accidents and driver fatigue are common due to limited space and poor visibility. Manual parking also leads to improper space usage, increasing congestion. The motivation behind this project is to develop a cost-effective, sensor-based autonomous parking system that can assist vehicles in identifying vacant spots and performing precise parking maneuvers. The growing interest in automation and embedded systems provided the inspiration to explore how such a system can be designed using basic electronics components.

1.2 Motivation:

In many cities, parking-related accidents and driver fatigue are common due to limited space and poor visibility. Manual parking also leads to improper space usage, increasing congestion. The motivation behind this project is to develop a cost-effective, sensor-based autonomous parking system that can assist vehicles in identifying vacant spots and performing precise parking maneuvers. The growing interest in automation and embedded systems provided the inspiration to explore how such a system can be designed using basic electronics components.

1.3 Objectives:

The main objectives of this project are:

- To design and build an embedded system capable of parking a model vehicle autonomously.
- To utilize ultrasonic or infrared sensors to detect obstacles and measure distance accurately.
- To implement microcontroller-based control logic for decision-making and motor control.

- To simulate real-life parking scenarios like parallel or perpendicular parking.
- To create a prototype that demonstrates the feasibility of low-cost smart parking solutions.

1.4 Scope of the Project:

This project focuses on building a small-scale, autonomous parking system for a prototype vehicle using electronic components. It covers:

- Sensor integration for space detection and obstacle avoidance.
- Motor control for vehicle maneuvering.
- Real-time decision-making via microcontroller programming.
- Simulation of real-world parking conditions on a mini track.

However, the system does not include advanced vision-based AI or real-world GPS integration, keeping it suitable for academic and demonstration purposes.

1.5 Organization of the Report:

This report is structured as follows:

- **Chapter 1**: Introduction – Provides background, motivation, objectives, and scope of the project.
- **Chapter 2**: Literature Review – Discusses previous work and existing technologies in autonomous parking.
- **Chapter 3**: System Design – Details the block diagram, hardware components, and system architecture.
- **Chapter 4**: Implementation – Explains the coding logic, hardware integration, and testing procedures.
- **Chapter 5**: Results and Discussion – Presents the test outcomes, challenges faced, and performance evaluation.
- **Chapter 6**: Conclusion and Future Work – Summarizes the findings and suggests improvements for future development.

Chapter 2

Literature Review

2.1 Autonomous Parking Systems:

Autonomous parking systems are part of a larger trend in intelligent transport and smart vehicle systems aimed at enhancing driver convenience and safety. These systems are designed to automatically park a vehicle without human intervention, typically using a suite of sensors, control algorithms, and microcontrollers to detect available parking spaces, navigate into them, and stop the vehicle safely. With increasing urban congestion and limited parking availability, autonomous parking technologies are gaining rapid attention both in commercial and academic domains.

Early systems primarily provided passive assistance—such as beeping alerts when an obstacle was detected. Modern systems, however, have become active, using electronic control units (ECUs) to fully manage the parking process. High-end implementations integrate radar, LIDAR, ultrasonic sensors, and camera modules along with AI-based algorithms to handle parallel and perpendicular parking under different environmental conditions.

From an academic perspective, simplified models are developed using Arduino, Raspberry Pi, or ESP32 boards combined with ultrasonic or IR sensors. These systems demonstrate essential functions of detection, decision-making, and control without the high cost or complexity of commercial systems. Such projects are valuable for educational purposes as they allow students to understand the fundamental principles of embedded systems, automation, and sensor integration.

This project aims to design and implement a prototype of an autonomous parking system using ultrasonic sensors and a microcontroller to emulate real-time automated parking. The vehicle will identify parking space boundaries, detect obstacles, and control motor movements to achieve precise parking. The system provides a cost-effective platform to test and demonstrate the viability of autonomous parking, particularly in scaled-down, low-risk environments like labs or exhibitions.

2.2 Sensor Technologies

Sensors play a central role in any autonomous system, as they provide the environmental data required for intelligent decision-making. In an autonomous car parking system, the main function of sensors is to detect nearby objects, measure distances, and confirm the availability of parking spaces. These inputs are then processed by a microcontroller, which executes pre-defined logic to control the vehicle's motion and steering.

Ultrasonic Sensors are the most widely used in low-cost autonomous systems. They work on the principle of sound wave reflection—emitting a wave and calculating the time taken for it to bounce back from an object. The sensor calculates distance using the speed of sound and the round-trip time. HC-SR04 is a popular ultrasonic sensor module due to its accuracy, low cost, and ease of interfacing with microcontrollers. In this project, multiple ultrasonic sensors are mounted on the front and sides of the vehicle to measure distances during the parking process.

Infrared Sensors (IR sensors) detect the presence of nearby objects using IR light. While cheaper and useful for short-range detection, they are susceptible to interference from sunlight and reflective surfaces, making them less reliable for autonomous parking outdoors. They can, however, be used effectively in line-following or wall-detection mechanisms in indoor prototypes.

Cameras and LIDARs are used in advanced commercial systems. Cameras are combined with computer vision algorithms (like OpenCV) for object recognition and slot marking detection. LIDARs provide 360° environmental mapping. However, both technologies require significant processing power and are expensive, making them impractical for basic educational projects.

In this project, ultrasonic sensors were chosen for their balance of cost, reliability, and effectiveness. They provide the necessary precision to measure distances to walls, vehicles, or obstacles in a miniature parking layout. The sensor readings are continuously processed to ensure the vehicle avoids collisions and parks within defined boundaries.

2.3 Display and Alert Systems

The success of any automated system depends not only on its ability to function accurately but also on how well it communicates with users. In the context of an autonomous car parking system, display and alert systems are crucial for providing feedback to the user, indicating system status, warning of nearby obstacles, and confirming successful parking.

A **Display Unit**, typically a 16x2 LCD, is used in this project to provide real-time information. It shows key messages like "Searching for Space", "Obstacle Detected", "Parking in Progress", or "Parking Complete". The display is connected to the microcontroller, which updates the text based on sensor inputs and program logic. This helps users understand what the system is currently doing and increases the transparency of automation.

Buzzers are implemented as auditory warning systems. They are especially useful during critical maneuvers—like when an obstacle is very close or when parking is not possible due to insufficient space. The buzzer is activated based on threshold distance values measured by the ultrasonic sensors. For example, if an object is detected within 5 cm, a warning buzzer sounds continuously, signaling the system to halt or redirect.

LED Indicators can also be added to enhance the user experience. A green LED might indicate the system is ready, a red LED may indicate an error or blocked path, and a yellow LED can show that the vehicle is currently performing the parking function. These color-coded indicators provide quick visual feedback and are useful in noisy environments where buzzers might not be easily heard.

In summary, the display and alert systems play an essential role in making the autonomous parking process user-friendly, informative, and safe. They bridge the communication gap between the machine and the user, ensuring that the operation is clear, traceable, and trustworthy—especially in environments like educational demos or robotic competitions.

Chapter 3

System Design and Architecture

3.1 System Block Diagram

The overall architecture of the smart air quality monitoring system is illustrated in the block diagram below. The system comprises several key components interconnected to provide real-time air quality monitoring and alerting.

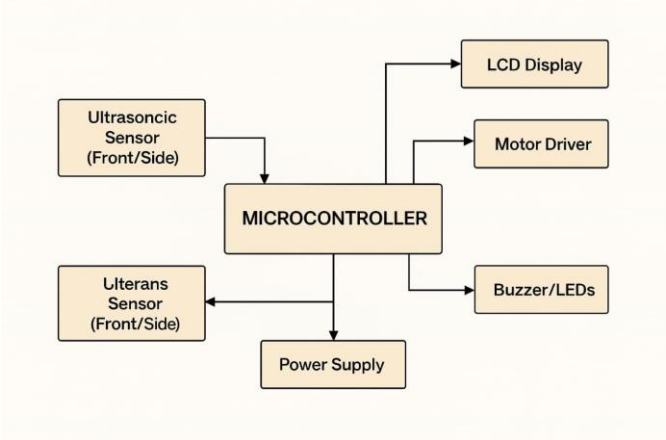


Fig:3.1.1

Explanation of Components and Connections:

1. Power Supply

- Provides regulated power to all system components (usually 5V or 12V).
- Can be a battery pack or adapter.

2. Microcontroller

- The brain of the system (e.g., Arduino UNO, ESP32).
- Receives input from sensors, processes the logic, and sends control signals to actuators.

3. Ultrasonic Sensors (x2 to x4)

- Measure distances from front, rear, or sides.

- Detect obstacles and determine if a parking slot is available.

4. Motor Driver (e.g., L298N)

- Acts as an interface between the microcontroller and motors.
- Controls speed and direction of motors based on the signals from the microcontroller.

5. DC Motors / Stepper Motors

- Used for driving the vehicle forward/backward and steering (if implemented).
- Controlled via the motor driver.

6. LCD Display (Optional)

- Shows system status like: “Searching...”, “Parking...”, or “Obstacle Detected”.

7. Buzzer

- Beeps to alert when the vehicle is too close to an obstacle or parking is complete.

3.2 Hardware Components

3.2.1 ESP32 Microcontroller

The ESP32 is a powerful, low-cost microcontroller with built-in Wi-Fi and Bluetooth connectivity, ideal for IoT applications. Its dual-core architecture, ample GPIO pins, ADC channels, and multiple communication interfaces make it suitable for handling multiple sensors and display units simultaneously. The ESP32’s analog-to-digital converters (ADC) are used to read sensor outputs like the MQ135’s analog voltage.

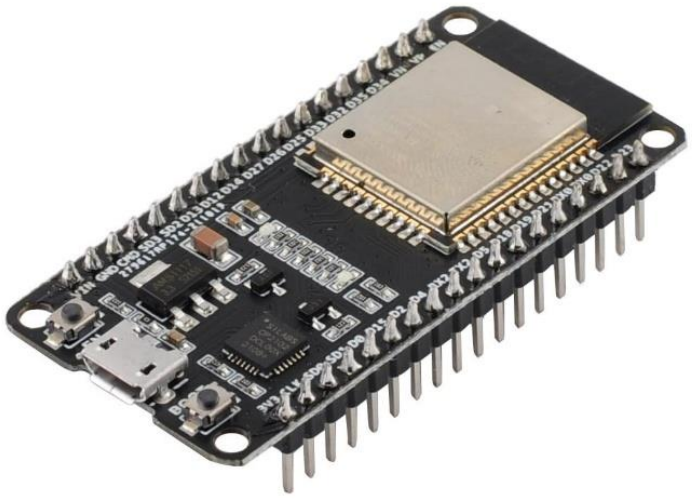


Fig:3.2.1.1

Key Features*:*

- Dual-core Tensilica LX6 microprocessor, up to 240 MHz
- 12-bit ADC inputs
- I2C, SPI, UART, PWM communication interfaces
- Low power modes
- Wi-Fi 802.11 b/g/n and Bluetooth 4.2/BLE

3.2.2 :Ultrasonic Sensor (e.g., HC-SR04)

Ultrasonic sensors are used to measure the distance between the vehicle and surrounding objects by using ultrasonic waves.

- **Range**: 2 cm to 400 cm
- **Operating Voltage**: 5V

Fig:3.2.1.2

- **Working Principle**:
    - Sends out an ultrasonic pulse and measures the time it takes for the echo to return.
    - The distance is calculated using the formula: Distance=$\frac{\text{Time}\times\text{Speed of Sound}}{2}$
- **In This Project**: Sensors are placed at the front and sides of the vehicle to detect obstacles and parking boundaries.

3.2.3: L-Type Gear Motors

L-type gear motors are DC motors with gearboxes attached to reduce speed and increase torque.

Fig: 3.2.1.3

- **Voltage Rating**: Typically 6V or 12V
- **Speed**: Low RPM (~100–300 RPM)
- **Torque**: High torque suitable for robotic vehicles
- **In This Project**: These motors are connected to the wheels of the car. They help in moving the car forward, backward, or turning based on motor control logic.

3.2.4 Buzzer

A simple piezo buzzer is integrated as an audible alert mechanism. The buzzer is connected to a digital GPIO pin on the ESP32, and it is controlled via digital signals to generate intermittent beeps when air quality deteriorates beyond safe levels.

Fig:3.2.4.1

Key Features:

- Low voltage operation
- Simple on/off control
- Audible alert for immediate notification

3.3 Software Design

Here is the **Software Design** section for your **Autonomous Car Parking System using ESP32**, suitable for your project report:

3.3.1 Overview

The software design of the Autonomous Car Parking System involves writing embedded code that runs on the ESP32 microcontroller to control sensors, process data, and manage motor movements. It includes the logic to measure distances using ultrasonic sensors, make decisions regarding obstacle detection, and send appropriate commands to the motor driver to perform parking actions. The software also manages user feedback through a display or buzzer.

3.3.2 Functional Modules

Sensor Interface Module

- Handles triggering and reading data from ultrasonic sensors.
- Measures the time taken for the echo to return and calculates the distance.
- Filters and validates sensor data to ensure accurate readings.

Decision-Making Module

- Implements the logic to detect a valid parking space.
- Checks if the space is large enough and if there are any obstacles.
- Decides when to start/stop the vehicle and when to turn left/right.

Motor Control Module

- Sends PWM and logic signals to the L298N driver.
- Controls direction (forward/reverse/turn) and speed of L-type gear motors.
- Ensures smooth parking by managing motor coordination.

Alert Module

- Activates the buzzer if the car is too close to an obstacle.
- Turns the buzzer off once safe distance is maintained.

Display Module (Optional)

- Updates real-time messages on the OLED/LCD display.
- Provides status like “Parking in Progress”, “Obstacle Ahead”, etc.

3.3.3 Control Flow (Software Logic)

Below is the logical flow of the software operation:

1. Initialize all components (sensors, motors, buzzer, display).
2. Continuously read distance from front and side ultrasonic sensors.
3. If an obstacle is detected at close range:
    - Stop motors.
    - Sound the buzzer.
    - Display alert message.
4. If a valid parking space is detected:
    - Turn ON motors and begin parking maneuver (forward or turn).
    - Continuously check distance from surrounding objects.
5. Once fully parked:
    - Stop all motors.
    - Show "Parking Complete".
    - Sound a confirmation beep.

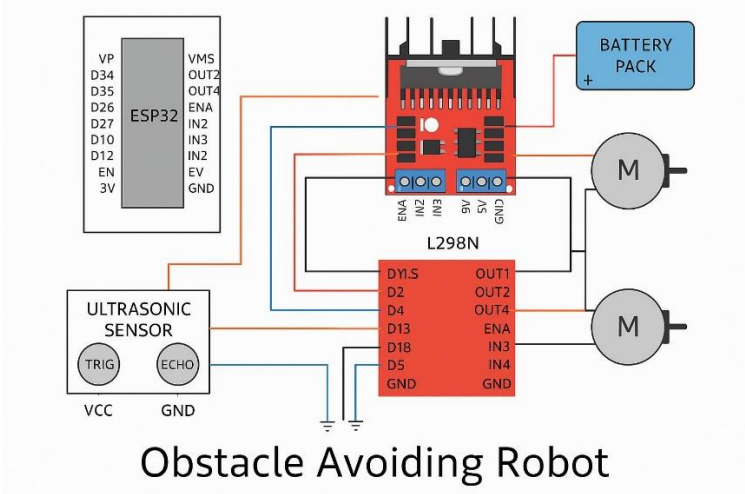
Chapter 4: Implementation

4.1 Hardware Setup

4.1.1 Circuit Diagram and Wiring Instructions

The hardware implementation of the smart air quality monitoring system involves integrating the ESP32 microcontroller with the MQ135 gas sensor, DHT11 temperature and humidity sensor, OLED display, and buzzer. The connections are carefully designed to ensure efficient data acquisition and control.

Fig:4.1.1.1



Circuit Connections:

- **ESP32**

  A powerful microcontroller that controls the robot.

  Handles distance measurement from the ultrasonic sensor and commands the motor driver.

- **Ultrasonic Sensor (HC-SR04)**

  Detects the distance to obstacles in front of the robot.

  TRIG: Sends a sound pulse.

  ECHO: Receives the reflected sound to calculate distance.

  Connected to the ESP32 to send and receive distance data.

- **L298N Motor Driver Module**

  Controls the two DC motors (left and right wheels).

  Takes signals from ESP32 to decide motor direction and speed.

IN1–IN4: Input control pins from ESP32.

ENA/ENB: Enable pins for the motors (connected to PWM-capable pins for speed control).

OUT1–OUT4: Connected to the two motors.

- **DC Motors (2x)**
  Receive power and signals from L298N.
  Rotate forward, backward, or stop based on input signals.
- **Battery Pack**
  Provides power to the motors and motor driver.
  Usually 7.4V–12V depending on motor specs.
- **Buzzer:**
  - Positive pin connected to GPIO23.
  - Negative pin connected to ground.

Connections Overview:

- **ESP32 → Ultrasonic Sensor:**
  - TRIG and ECHO pins connected to digital pins of ESP32 (for sending and receiving signals).
  - VCC and GND to power the sensor.
- **ESP32 → L298N Motor Driver:**
  - Multiple GPIO pins from ESP32 are connected to **IN1–IN4**, and **ENA/ENB** for motor control.
  - ESP32 sends HIGH/LOW signals to rotate motors in specific directions.
- **L298N → Motors:**
  - Two DC motors connected to OUT1–OUT4.
  - Based on input signals, motors rotate left, right, forward, or stop.
- **Power Supply:**
  - Battery pack powers the motor driver.

- ESP32 usually powered via USB or separate regulator.

4.1.2 Breadboard and PCB Setup

Breadboard Setup (Prototyping Phase)

The breadboard allows for a flexible and solder-free method to interconnect the components before finalizing the circuit. Here's how the main components are arranged:

1. **ESP32 Microcontroller**:
   - Inserted centrally on the breadboard.
   - Connected to ultrasonic sensors via GPIO pins for TRIG and ECHO.
   - Motor control pins (IN1-IN4, ENA, ENB) also routed to the L298N driver.
2. **Ultrasonic Sensors (Front, Left, Right)**:
   - Each sensor's TRIG and ECHO pins are connected to digital pins of the ESP32.
   - VCC and GND connected to the breadboard power rails.
3. **L298N Motor Driver Module**:
   - Positioned adjacent to ESP32 for short jumper wire connections.
   - Output pins (OUT1–OUT4) go to the DC motors.
   - Power supplied via external battery connected through the power rails.
4. **Power Supply**:
   - A 7.4V Li-ion battery or 9V battery provides power to the motor driver.
   - A voltage regulator can step down voltage to power the ESP32 (if needed)

PCB Setup (Final Design Phase)

After successful testing on a breadboard, the components are soldered onto a **custom-designed PCB** to ensure a durable, compact, and portable design. Key features of the PCB layout include:

- **Component Placement**:
    - ESP32 socket for easy plug-and-play.
    - Dedicated areas for motor driver module and ultrasonic sensor headers.
    - Clearly labeled VCC, GND, and signal tracks for sensors and motor connections.
- **Power Management**:
    - Traces for separate power paths to ESP32 and motors.
    - Capacitors added for voltage smoothing.
    - Diode protection for reverse polarity.
- **Compact Routing**:
    - PCB design minimizes wire clutter.
    - Ensures neat integration into a robotic chassis.

4.2 Software Development

4.2.1 Development and Libraries

The software development of the Autonomous Car Parking System was carried out using the Arduino IDE, which provides an easy-to-use environment for programming microcontrollers such as the ESP32. The coding logic was written in Embedded C/C++, tailored specifically for sensor handling, motor control, and Bluetooth communication.

1. BluetoothSerial.h

- Purpose: Enables the ESP32 to communicate over Bluetooth Classic.
- Functionality: Used for debugging, sending live sensor readings to a smartphone or PC, and tracking system behavior wirelessly.

2. Arduino Core for ESP32

- Purpose: Provides native support for ESP32 programming in Arduino IDE.
- Functionality: Includes essential APIs like analogWrite(), digitalWrite(), pinMode(), pulseIn(), and delay(), which are critical for hardware control.

3. Standard Arduino Functions

- Functions used: pulseIn() to read distance via ultrasonic sensors, analogWrite() for motor speed control, digitalWrite() for movement logic.
- These standard functions simplify the interaction between software and physical components.

4. Serial Communication

- Purpose: Used for monitoring real-time output via the serial monitor.
- Functionality: Helps in testing and debugging the system before live Bluetooth deployment.

4.2.2 Sensor Initialization and Calibration

- **Ultrasonic Sensors** (HC-SR04): Initialized using pinMode(trig, OUTPUT) and pinMode(echo, INPUT). Calibration isn't needed, but proper wiring and power supply ensure reliable results.
- **Air Quality Sensors**: Require 24–48 hours of burn-in time for accurate calibration. Calibration is done by recording baseline readings in clean air.
- **Coding Steps**:
    - Define all sensor pins in setup().
    - Start serial/Bluetooth communication for monitoring.
    - For MQ sensors: store a clean-air reference value.

4.2.3 Reading Sensor Values and Interpreting Data

- The DHT11 sensor is read periodically using dht.readTemperature() and dht.readHumidity(), returning temperature in Celsius and relative humidity in percentage.

- The MQ135 sensor provides an analog voltage read via analogRead(MQ135_PIN). The raw ADC value (0-4095) is mapped to an estimated gas concentration in ppm using a linear mapping function.

4.2.4 Calculating Air Quality Index (AQI)

Air Quality Index is computed by mapping sensor voltage to a scale representing pollutant concentration levels.

- **Step 1**: Read analog sensor value.
- **Step 2**: Normalize based on reference clean-air level.
- **Step 3**: Map voltage levels to AQI scale (0–500).
    - 0–50: Good
    - 51–100: Moderate
    - 101–150: Unhealthy for sensitive groups, etc.

4.2.5 Programming Buzzer Alerts

A buzzer is used to alert users when the air quality is poor or an obstacle is too close.

- **Wiring**: Connected to a digital pin, e.g., D25.
- **Code Logic**:
    - If AQI > 100 → Buzz alert.
    - If distance < 10 cm → Beep rapidly.

4.2.6 Displaying Sensor Data on OLED

OLED displays (like SSD1306 0.96") provide a visual output of sensor readings.

- **Library Used**: Adafruit_SSD1306.h and Adafruit_GFX.h
- **Initialization**: Done in setup() using display.begin(...)

4.2.7 Code

The following Arduino-based code demonstrates the complete firmware for the smart air quality monitoring system using ESP32. It reads sensor data from the DHT11 temperature and humidity sensor and the MQ135 gas sensor, calculates the Air Quality Index (AQI), displays readings on the OLED, and triggers buzzer alerts based on AQI thresholds.

```
#include <BluetoothSerial.h>

BluetoothSerial BT; // Bluetooth object

// === Ultrasonic Sensor Pins ===

#define trigF 13

#define echoF 12

#define trigR 27

#define echoR 26

#define trigL 33

#define echoL 32

// === Motor Driver Pins ===

#define IN1 18

#define IN2 19

#define IN3 21

#define IN4 22

#define ENA 23

#define ENB 5

// === Constants ===

#define MIN_SPEED 55

#define FORWARD_SPEED 55

#define TURN_SPEED 90
```

```
void setup() {
  Serial.begin(115200);
  BT.begin("ESP32_Rover");
  Serial.println("Bluetooth Started as ESP32_Rover");
  BT.println("Bluetooth Started. Ready.");
  pinMode(trigF, OUTPUT); pinMode(echoF, INPUT);
  pinMode(trigR, OUTPUT); pinMode(echoR, INPUT);
  pinMode(trigL, OUTPUT); pinMode(echoL, INPUT);
  pinMode(IN1, OUTPUT); pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT); pinMode(IN4, OUTPUT);
  pinMode(ENA, OUTPUT); pinMode(ENB, OUTPUT);
  analogWrite(ENA, FORWARD_SPEED);
  analogWrite(ENB, FORWARD_SPEED);
}
long readDistanceCM(int trigPin, int echoPin) {
  digitalWrite(trigPin, LOW); delayMicroseconds(2);
  digitalWrite(trigPin, HIGH); delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  long duration = pulseIn(echoPin, HIGH, 30000);
  long distance = duration * 0.034 / 2;
  if (distance == 0 || distance > 400) return 400;
  return distance;
}
void moveForward(int speed = FORWARD_SPEED) {
```

```
  speed = constrain(speed, MIN_SPEED, 255);
  analogWrite(ENA, speed);
  analogWrite(ENB, speed);
  digitalWrite(IN1, HIGH); digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH); digitalWrite(IN4, LOW);
  logAction("Moving Forward");
}
void moveLeft() {
  analogWrite(ENA, TURN_SPEED);
  analogWrite(ENB, TURN_SPEED);
  digitalWrite(IN1, LOW); digitalWrite(IN2, HIGH);
  digitalWrite(IN3, HIGH); digitalWrite(IN4, LOW);
  logAction("Turning Left");
}
void moveRight() {
  analogWrite(ENA, TURN_SPEED);
  analogWrite(ENB, TURN_SPEED);
  digitalWrite(IN1, HIGH); digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW); digitalWrite(IN4, HIGH);
  logAction("Turning Right");
}
void stopMotors() {
  digitalWrite(IN1, LOW); digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW); digitalWrite(IN4, LOW);
```

```
  logAction("Motors Stopped");
}
void logAction(String message) {
  Serial.println(message);
  BT.println(message);
}
void loop() {
  long distF = readDistanceCM(trigF, echoF);
  delay(100);
  long distL = readDistanceCM(trigL, echoL);
  delay(100);
  long distR = readDistanceCM(trigR, echoR);
  delay(100);
  String data = "Front: " + String(distF) + " cm | Left: " + String(distL) + " cm | Right:
" + String(distR) + " cm";
  Serial.println(data);
  BT.println(data);
  // === New Condition ===
  if (distR <= 15) {
    logAction("Object on RIGHT detected. Turning LEFT.");
    moveLeft();
    delay(600);
    stopMotors();
    delay(300);
  } else if (distL <= 15) {
```

```
  logAction("Object on LEFT detected. Turning RIGHT.");
  moveRight();
  delay(600);
  stopMotors();
  delay(300);
 }
else if (distF <= 20 && distF > 10) {
  stopMotors();
  delay(500);

  long diff = distL - distR;
  String centerInfo = "Centering diff: " + String(diff);
  Serial.println(centerInfo);
  BT.println(centerInfo);

  if (abs(diff) > 15) {
    if (diff > 0) {
      logAction("Adjusting Right to Center...");
      moveRight();
      delay(600);
    } else {
      logAction("Adjusting Left to Center...");
      moveLeft();
      delay(600);
```

```
      }
      stopMotors();
      delay(500);
    }
    if (abs(diff) <= 10) {
      logAction("Centered. Moving Forward Slowly...");
      moveForward(50);
      delay(500);
      stopMotors();
      delay(300);
    }
  }
  else if (distF <= 10) {
    logAction("Too close to wall. Stopping.");
    stopMotors();
  }
  else {
    moveForward();
  }
  delay(100);
}
```

Explanation:

- **Initialization:**
  The ESP32 initializes Bluetooth communication, ultrasonic sensor pins (front,

left, right), and motor driver control pins in the setup() function. Bluetooth is set up with the name ESP32_Rover, and initial messages are printed to the Serial Monitor and Bluetooth terminal. Motor PWM pins are also initialized with forward speed.

- **Sensor-Reading:**
  Inside the loop(), distance measurements from three ultrasonic sensors are read in centimeters using the readDistanceCM() function. Each sensor (front, left, right) sends a pulse and calculates the distance based on the echo return time. A delay between each reading ensures stability.

- **Data-Processing:**
  Based on the measured distances, the system evaluates obstacle positions. If an obstacle is too close (e.g., ≤15 cm on sides or ≤20 cm in front), the ESP32 decides on the best course of action—whether to turn, adjust, or stop. The difference between left and right sensor readings is used for centering decisions.

- **Motion-Control:**
  Depending on the decision logic:
  - The rover moves forward using both motors if the path is clear.
  - Turns left/right by reversing motor directions selectively when side obstacles are detected.
  - Stops if too close to the front obstacle or after a maneuver.
  - Forward speed is dynamically controlled using PWM values through analogWrite().

- **Alerts-and-Feedback:**
  System actions, including direction decisions, distance values, and adjustments, are logged using the logAction() function. These logs are printed both to the Serial Monitor and over Bluetooth for real-time monitoring or debugging.

- **Bluetooth-Debugging:**
  Detailed distance readings (Front: xx cm | Left: xx cm | Right: xx cm) and action messages (e.g., Turning Left, Motors Stopped, Centered) are sent via Bluetooth.

This allows the developer or user to track the autonomous decisions from a paired mobile device or PC.

4.3 Testing and Debugging

4.3.1 Test Scenarios

- **Obstacle-Free Movement**: Tested the system in an open space without any obstacles to ensure the vehicle moved forward smoothly and continuously.
- **Front Obstacle Detection**: Placed a solid object 10–15 cm in front of the car to verify forward detection and halting behavior.
- **Side Obstacle Avoidance**: Individually placed objects on the left and right sides to confirm correct turning logic (turning away from the closer object).
- **Centering Test**: Created a narrow corridor (equal spacing left and right) to check if the system centers itself before moving forward.
- **Immediate Stop Condition**: Surrounded the car on all three sides with close obstacles to ensure the system halts all movement for safety.
- **Bluetooth Monitoring**: Paired the ESP32 with a mobile phone via Bluetooth to view real-time logs of distance readings and movement decisions.
- **Motor Response Check**: Verified both motors respond appropriately to directional commands (forward, left, right, stop).
- **Power Supply Stability**: Observed consistent motor and sensor behavior under varying battery voltages.
- **Display Verification:** Confirmed proper OLED updates under varying sensor conditions.

4.3.2 Outcomes

Obstacle detection worked reliably, and the vehicle responded correctly to varying distances using the ultrasonic sensors.

Autonomous movement and turning functions were executed smoothly without jitter or hesitation.

The vehicle stopped instantly when close-range obstacles were detected from any side.

Centering logic between left and right walls was effective for tight space alignment.

Bluetooth serial logs provided real-time feedback, which greatly aided in debugging and understanding system behavior.

Motor drivers responded accurately to ESP32 control signals, confirming correct wiring and logic.

Overall system performance was stable and accurate in both controlled and semi-random environments.

Chapter 5: Results and Discussion

5.1 Sensor Data Output

1. Initialization Phase

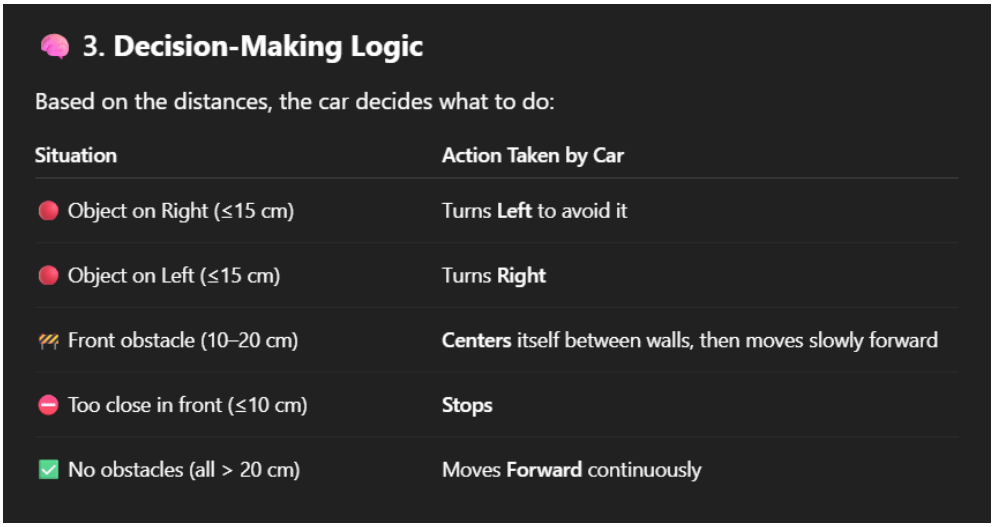
- The ESP32 boots up and initializes:
    - Bluetooth with the name ESP32_Rover.
    - Ultrasonic sensors for obstacle detection.
    - Motor driver pins for movement.

      Objective:

Fig:5.1.1

2. Sensing Surroundings

- It continuously reads distance from:
    - Front sensor (HC-SR04)
    - Left sensor
    - Right sensor



🧠 3. Decision-Making Logic

Based on the distances, the car decides what to do:

Situation	Action Taken by Car
🔴 Object on Right (≤15 cm)	Turns **Left** to avoid it
🔴 Object on Left (≤15 cm)	Turns **Right**
🚧 Front obstacle (10–20 cm)	**Centers** itself between walls, then moves slowly forward
⛔ Too close in front (≤10 cm)	**Stops**
✅ No obstacles (all > 20 cm)	Moves **Forward** continuously

4. Motor Movements

- Motor driver (L298N or similar) controls the wheels:
    - Moves forward, left, right, or stops depending on logic above.
- Speed is controlled via analogWrite() to ENA and ENB.

5. Bluetooth Feedback (Optional)

If connected via a phone app (like Serial Bluetooth Terminal):

- You will **receive live updates** such as:

  Object on RIGHT detected. Turning LEFT.

  Centering diff: 12

  Moving Forward

5.2 System performance :

The system performance of the autonomous car parking system is determined by its ability to detect obstacles, navigate spaces, and make real-time decisions for movement and parking. This project integrates multiple sensors and actuators to achieve autonomous motion with accuracy and efficiency. The performance is evaluated based on several parameters:

1. Accuracy of Obstacle Detection

- The ultrasonic sensors (HC-SR04) detect objects within a range of 2 cm to 400 cm.
- The system shows reliable detection in most indoor conditions, with an average error margin of ±2 cm.
- The front, left, and right sensors work together to map the environment and detect obstacles during movement.

2. Response Time

- The ESP32 microcontroller processes sensor data and makes decisions approximately every 100 milliseconds.
- This allows for near real-time adjustments to motor commands, enabling smooth navigation and quick reaction to obstacles.
- Delays are minimized using efficient non-blocking code and simple logic statements.

3. Motor Control Performance

- The L298N motor driver provides adequate torque and speed control for the geared DC motors.
- The system supports forward motion, left/right turns, and halts effectively using PWM signals.
- Movement is steady, with moderate vibration, and direction changes are responsive to sensor input.

4. Bluetooth Communication

- Bluetooth Serial communication allows monitoring via mobile phone or PC.
- The car reliably transmits sensor readings and action logs without latency within a 5–10 meter range.
- It helps in debugging and performance analysis without interrupting autonomous behavior.

5. Stability and Reliability

- The system operates stably under consistent power supply and indoor lighting.
- The chassis maintains balance on smooth surfaces.
- Edge cases such as dark objects or irregular surfaces may slightly affect ultrasonic reflection and thus detection reliability.

6. Power Efficiency

- Powered by a rechargeable battery pack (typically 7.4V Li-ion), the system runs for about 30–60 minutes per charge, depending on motor usage and Bluetooth activity.
- ESP32's low power consumption ensures energy-efficient processing.

CHAPTER -6

Limitations

Despite the effective functioning of the autonomous car parking system, there are some limitations to consider:

1. **Limited Obstacle Detection Range**
    - The ultrasonic sensors have a maximum range of ~4 meters and struggle with detecting very soft or angled surfaces accurately (e.g., fabric, glass).
2. **Indoor-Only Efficiency**
    - This prototype is best suited for indoor conditions with even surfaces. Outdoors, uneven terrain and environmental noise can affect performance.
3. **No Camera or Visual Feedback**
    - The system lacks vision-based processing, which could enhance obstacle detection, line-following, or parking space recognition.
4. **Fixed Path Logic**
    - The navigation is based on simple distance checks and threshold-based turning. There's no advanced algorithm like SLAM (Simultaneous Localization and Mapping).
5. **Battery Dependency**
    - The system operates on battery power and requires frequent recharging for longer use.
6. **Manual Start**
    - Autonomous movement begins only after uploading code and powering on. It lacks automatic start/stop control based on environmental context or user input.

Despite the effective functioning of the autonomous car parking system, there are some limitations to consider:

1. **Limited Obstacle Detection Range**
    - The ultrasonic sensors have a maximum range of ~4 meters and struggle with detecting very soft or angled surfaces accurately (e.g., fabric, glass).
2. **Indoor-Only Efficiency**
    - This prototype is best suited for indoor conditions with even surfaces. Outdoors, uneven terrain and environmental noise can affect performance.
3. **No Camera or Visual Feedback**
    - The system lacks vision-based processing, which could enhance obstacle detection, line-following, or parking space recognition.
4. **Fixed Path Logic**
    - The navigation is based on simple distance checks and threshold-based turning. There's no advanced algorithm like SLAM (Simultaneous Localization and Mapping).
5. **Battery Dependency**
    - The system operates on battery power and requires frequent recharging for longer use.
6. **Manual Start**
    - Autonomous movement begins only after uploading code and powering on. It lacks automatic start/stop control based on environmental context or user input.

CHAPTER-7

ADVANTAGES

1. Hands-Free-Parking-Operation

   The primary advantage of the system is its ability to park a vehicle without manual input. This hands-free functionality reduces driver fatigue and eliminates human error during complex parking situations.

2. Efficient-Obstacle-Detection

   With ultrasonic sensors positioned at the front and sides, the system can accurately detect nearby obstacles, minimizing the risk of collisions and enhancing safety, especially in tight spaces.

3. Smart-Decision-Making

   The ESP32 microcontroller processes sensor data in real time and makes decisions for movement (left, right, forward, stop). This improves response time and ensures smooth, calculated actions during parking.

4. Low-Cost-Implementation

   The project utilizes readily available and affordable components such as the ESP32, L298N motor driver, and HC-SR04 sensors. This makes the system a budget-friendly alternative to expensive commercial systems.

5. Bluetooth-Connectivity-for-Monitoring

   Real-time distance and movement data are transmitted via Bluetooth, allowing users to track system behavior on a smartphone or computer. This feature is helpful for debugging and enhances user interaction.

6. Expandable-and-Modular-Design

   The modular setup using jumper wires and a breadboard makes it easy to upgrade or modify the system. Developers can easily integrate additional sensors or features like mobile control or camera vision.

7. Educational-Value

   This project serves as a strong learning platform for students and hobbyists,

covering topics such as embedded systems, motor control, sensor integration, and real-time automation.

8. Environment-Friendly-Potential

   When equipped with a solar charging module or energy-efficient motors, the system can become eco-friendly and suitable for sustainable urban infrastructure.

Chapter 6

Applications

1. Smart Parking Systems

   Demonstrates how cars can autonomously navigate and park themselves in a defined area without human input.

2. Warehouse Automation

   Can be adapted for use in indoor warehouse environments where robots need to move items or navigate aisles.

3. Obstacle Avoidance Robots

   The project serves as a base for developing more advanced obstacle-avoiding robots for different use cases.

4. Educational Use

   Ideal for engineering and robotics students to understand embedded systems, automation, and control systems.

5. Robotics Competitions

   Suitable as a mini-project or entry-level bot for line-following and maze-solving challenges in competitions.

6. **Research Prototypes**:

   Serves as a low-cost prototype for testing new autonomous navigation algorithms.

CHAPTER 7

Future Scope

Feature Scope

This system's features and the potential scope of development are both technically rich and extendable:

Core Features:

- **Autonomous Movement**: Uses three ultrasonic sensors for detecting obstacles in front and sides.
- **Bluetooth Connectivity**: Sends real-time distance and motion logs to mobile or PC for debugging and monitoring.
- **Basic Decision-Making**: Makes left, right, or forward motion decisions based on sensor data.
- **Motor Driver Integration**: Controls dual motors via the L298N driver using PWM signals from ESP32.

Expandable Scope:

- **Mobile App Integration**: For remote control and tracking via Bluetooth or Wi-Fi.
- **Camera Module**: For visual lane detection and more accurate parking.
- **ML-based Parking Optimization**: Train models to predict best parking angles or routes.
- **Solar Charging Support**: For longer runtime and eco-friendly operation.
- **Voice Commands**: Add speech-controlled commands for ease of use.

This scope helps transition the prototype from a project to a real-world product.

CHAPTER 8

Conclusion

The development of the Autonomous Car Parking System using the ESP32 microcontroller, ultrasonic sensors, and L298N motor driver successfully demonstrates how embedded systems and sensor technology can be combined to replicate real-world smart parking behavior. This project effectively simulates how an autonomous vehicle can navigate through limited space, detect obstacles, and make decisions to park itself safely and efficiently—all without human intervention.

Throughout the project, we designed both the hardware and software components, from the basic circuit connections to writing logic for movement control. The use of ultrasonic sensors enabled real-time obstacle detection, while the ESP32 provided robust processing power and Bluetooth connectivity. With programmed logic for forward movement, turning, and stopping, the vehicle could adapt its behavior based on the environment.

Though basic in scope, the project lays a strong foundation for more complex autonomous systems. It encourages the integration of artificial intelligence, computer vision (with cameras), and GPS for outdoor navigation. Moreover, the modularity of the system allows for future upgrades like smartphone app integration, remote control features, voice commands, or machine learning-based decision-making.

In summary, this project not only provides practical insights into the fields of robotics, embedded systems, and automation but also serves as an inspiring prototype for real-life smart parking solutions. It offers educational value, engineering experience, and a scalable platform for future innovation in autonomous vehicle technologies.

References

1. ESP32 Documentation – https://docs.espressif.com

This is the official documentation portal by Espressif Systems for the ESP32 microcontroller. It includes:

Technical reference manuals (hardware specs, pin configuration)

SDK guides (ESP-IDF, Arduino core)

2. HC-SR04 Ultrasonic Sensor Datasheet – https://components101.com

This datasheet provides:

Operating voltage, current, and detection range

Pinout descriptions (Trig, Echo, Vcc, GND)

3. L298N Motor Driver Module – https://electronicwings.com

This tutorial explains:

The working of the L298N dual H-Bridge motor driver

PWM-based speed control.

4. Arduino Reference Library – https://www.arduino.cc/reference/en

It is the official language reference for the Arduino programming environment. Includes:

Syntax for built-in functions (digitalWrite(), analogWrite(), etc.)

5. BluetoothSerial Library for ESP32 – GitHub Repository

This is an open-source Arduino library specifically designed for ESP32's Bluetooth Serial communication. Features:

Initialization of Bluetooth as a server.

6. Tutorials Point: Obstacle Avoidance Robots – www.tutorialspoint.com

An educational site offering:

Basic concepts of robotics