International Journal of Science, Strategic Management and Technology
Volume **02** Issue **04** April-**2026** | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

Automatic Car Parking System by using ESP 32

Aryan Kuchekar College: Trinity Polytechnic Pune.
Pravin Saware Guide: **Mrs. R.P. KADGI**
Mahesh Sawant
Email**:** sunita06kuchekar@gmail.com Email: rsrutuja@gmail.com

https://doi.org/10.55041/ijsmt.v2i4.041

Cite this Article: Kuchekar, A., Saware, P. & Sawant, M. (2026). Automatic Car Parking System by using ESP 32. International Journal of Science, Strategic Management and Technology, *02*(04). https://doi.org/10.55041/ijsmt.v2i4.041

ABSTRACT

The primary objective of modern urban infrastructure is to enhance efficiency and minimize human error through seamless automation. The concept of "Intelligent Parking" is driven by rapid advancements in the Internet of Things (IoT) and real-time data processing. In this context, Microcontroller-based sensing technology plays a vital role, acting as the interface between physical vehicle movement and digital management systems. By utilizing the ESP32 microcontroller, this project provides a robust, hands-free solution for parking management that eliminates the need for manual oversight. Unlike traditional parking methods, this system employs ultrasonic sensors and infrared modules to detect slot availability in real-time, regardless of lighting conditions or human visibility.

KEYWORDS: *Embedded Systems, IOT (Internet of Things, AI (Artificial Intelligence).*

INTRODUCTION

The s global urbanization accelerates, the density of vehicles in metropolitan areas has reached an unprecedented scale. One of the most persistent challenges in modern "Smart Cities" is the inefficient management of parking spaces. Statistics suggest that a significant percentage of urban traffic congestion is caused by drivers searching for available parking, leading to wasted fuel, increased carbon emissions, and heightened driver frustration. Traditional parking systems, which rely on manual ticketing or human oversight, are no longer sufficient to handle the dynamic requirements of high-traffic environments. The emergence of the Internet of Things (IoT) has revolutionized how we interact with physical infrastructure. At the heart of this revolution is the ESP32 microcontroller, a powerful, low-cost system-on-a-chip (SoC) with integrated Wi-Fi and dual-mode Bluetooth. Its ability to process sensor data and communicate wirelessly in real-time makes it an ideal candidate for automating the parking process. By bridging the gap between hardware sensors and cloud-based data management, the ESP32 transforms a static parking lot into an "intelligent" ecosystem.

In the realm of embedded systems, selecting the right controller is critical for scalability. While previous systems relied on the Arduino Uno or basic 8-bit processors, the ESP32 (System on a Chip) represents a generational leap.

- Dual-Core Processing: Unlike single-core alternatives, the ESP32 can handle sensor data acquisition on one core while managing Wi-Fi stack communication on the other, preventing system "lag" during high vehicle throughput.
- Low Power Consumption: With advanced power management and "Deep Sleep" modes, the system can operate on battery or solar power in outdoor parking lots, significantly reducing the carbon.

International Journal of Science, Strategic Management and Technology
Volume **02** Issue **04** April-**2026** | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

LITERATURE SURVEY

To draft a Literature Survey for an Automatic Car Parking System using ESP32, you need to highlight what has been done before (using RFID, Arduino, or basic sensors) and identify the "gap" that your project fills (IoT integration, real-time cloud data, or Machine Learning). Following your style, here is a professional Literature Survey for your project: To draft a Literature Survey for an Automatic Car Parking System using ESP32, you need to highlight what has been done before (using RFID, Arduino, or basic sensors) and identify the "gap" that your project fills (IoT integration, real-time cloud data, or Machine Learning). Following your style, here is a professional Literature Survey for your project:

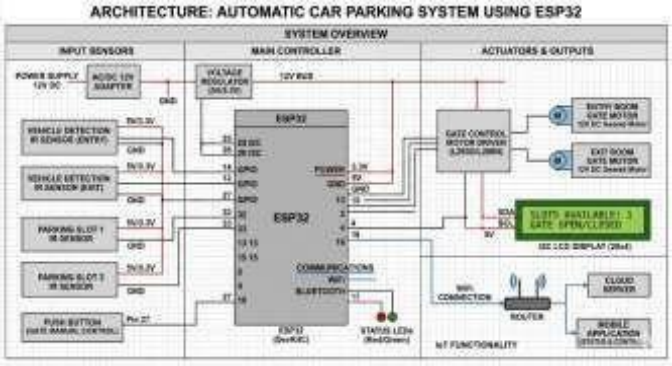
A significant portion of existing literature focuses on the use of Arduino Uno combined with Ethernet shields.

While these studies proved that ultrasonic sensors could accurately detect vehicle presence, they faced two major hurdles:

- Connectivity Constraints: Arduino-based systems often require bulky external modules for Wi-Fi, increasing the physical footprint and power consumption of the device.
- Static Interfaces: Many older projects utilized simple LCD screens at the entrance. As noted in recent critiques of home automation, relying solely on stationary displays restricts the user's ability to plan their arrival in advance.

In contrast, recent studies have begun to favor the ESP32 SoC. Research by suggests that the ESP32's integrated Wi-Fi stack allows for a "Cloud-First" approach, where data is not just displayed locally but is broadcast to a global web interface, solving the "omnipresence" issue mentioned in earlier home automation critiques. Current literature highlights a heavy reliance on single-sensor nodes. Some systems propose using only Infrared (IR) sensors; however, as documented in various technical reviews, IR sensors are highly susceptible to ambient light interference and dust, leading to "false positives" (reporting a spot as occupied when it is empty).

Architecture Diagram



PROPOSED SYSTEM

. The Proposed System features an advanced, automated parking management solution centered around the ESP32 microcontroller, designed to optimize vehicle flow and space utilization without human intervention. The architecture utilizes a dual-gate configuration, where two high-torque DC geared motors or servo motors act as independent entry and exit barriers. At the entrance, an infrared (IR) or ultrasonic sensor detects an approaching vehicle, triggering the ESP32 to verify slot availability via real-time monitoring of individual parking bay sensors. If a space is available, the entry motor activates the boom gate, and the system updates a local I2C LCD display to guide the driver. Simultaneously, an identical exit mechanism manages departing vehicles, using a secondary motor and sensor array to ensure a seamless flow and accurate incrementing of the available space count.

Leveraging the ESP32's integrated **Wi-Fi and Bluetooth stacks**, the system functions as an **IoT (Internet of Things)** node, transmitting occupancy data to a centralized cloud server or a mobile application. This connectivity allows users to check for available spots remotely and enables administrators to monitor gate status or manual overrides via a web dashboard. For enhanced reliability, the system incorporates a dedicated power management circuit to provide stable voltage to both the logic controllers and the high-draw actuators. By integrating automated sensing, mechanical control, and wireless data logging, the proposed system minimizes traffic congestion, reduces carbon emissions from idling vehicles, and provides a scalable framework for modern smart city infrastructure.

International Journal of Science, Strategic Management and Technology
Volume **02** Issue **04** April-**2026** | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

IMPLEMENTATION

Implementing the Automatic Car Parking System using an ESP32 microcontroller involves several sequential stages, beginning with hardware selection and culminating in firmware deployment and optional IoT integration. The physical implementation requires constructing a rigid model or deploying to a full-scale gate mechanism. The ESP32 (DevKitC), acting as the central unit, is first connected to the dual high-torque DC geared motors or servo motors that drive the independent entry and exit boom gates. Since these motors consume significant current, they are not powered directly by the ESP32 but rather through a robust motor driver (like the L293D or L298N), which receives PWM signals from the microcontroller while drawing power from a stable 12V bus. A comprehensive sensor array must be installed to facilitate the logic flow. Infrared (IR) reflective sensors are typically deployed at the entrance and exit barriers to detect vehicle proximity. Further implementation details include placing individual sensors (often specialized ultrasonic or inductive loop detectors) in each designated parking bay to monitor slot occupancy. All these sensor inputs are connected to the ESP32's available GPIO pins, configured as pull-up or pull-down inputs as required. For user interaction, an I2C LCD display (e.g., 20x4) is connected to the specialized I2C pins (GPIO 25/26), providing real-time feedback such as available spaces, "Gate Open," or "System Full" status. The final crucial step in the basic implementation is uploading the firmware (written in C++ via Arduino IDE or PlatformIO), which manages the complex looping logic: waiting for entry detection, checking slot counts, opening the corresponding motor, waiting for the car to clear, closing the gate, and updating the count.

For a full implementation including the optional IoT functionality, the ESP32's integrated Wi-Fi stack must be configured. This involves programming the ESP32 to connect to a local network and establish a connection to an IoT cloud server (such as Firebase, AWS, or Azure IoT). Once connected, the firmware is modified to transmit slot status and gate activity logs (timestamps, available slots) using MQTT or HTTP POST. This allows the system state to be monitored and controlled from a mobile application or web dashboard, where administrators can view real-time logs and, if necessary, trigger a manual gate override, completing the automation cycle.

Fig: 1: Block Diagram

the system architecture for an ESP32-based Automatic Car Parking System. The system is built around an ESP32 Microcontroller (the central hub) which processes inputs from a multi-sensor array to control vehicle motion and provide user feedback.

The inputs consist of Ultrasonic Sensors (Front/Rear) and Infrared Sensors (Side), which provide critical Distance Data and Proximity Data, respectively, enabling the car to map its environment and detect obstacles for safe navigation. The system is activated or deactivated by a physical Push Button (Start/Stop).

Based on the sensor inputs, the ESP32 controls the vehicle's movement through two main actuator interfaces:

1. DC Geared Motors (Wheels), managed via a Motor Driver (H-Bridge) with an associated Relay Module, control the car's speed and propulsion.

2. A Servo Motor (Steering), receiving PWM (Pulse Width Modulation) signals directly from the For user interaction and monitoring, the system utilizes a 128x64 OLED Display, connected via an I2C Bus, which shows real-time information such as "PARKING: Slot 3" and "DIST: 25cm."

Furthermore, the ESP32's integrated Bluetooth Module (indicated by the Bluetooth icon) acts as a wireless communication link, allowing the same real-time data to be sent to and displayed on a remote Smartphone App / Cloud Platform for advanced user control or data logging. The entire system is powered by a stable Power Supply unit incorporating (12V/5V) Regulators to match the varying voltage requirements of the sensors, microcontroller, and motors.

International Journal of Science, Strategic Management and Technology
Volume **02** Issue **04** April-**2026** | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

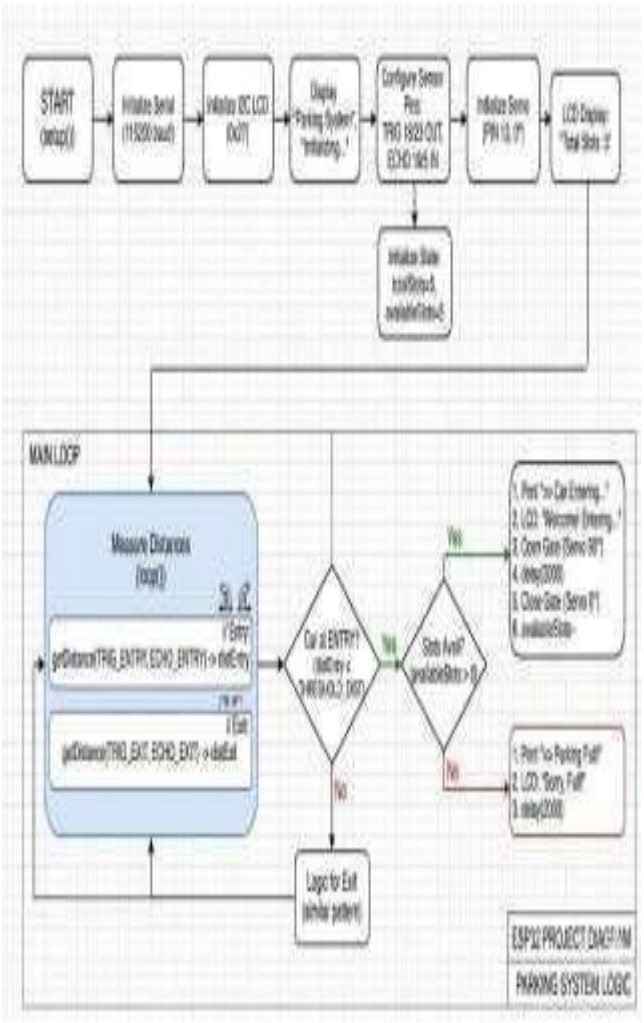
.HARDWARE IMPLEMENTATION

Implementing a virtual assistant-based smart home control system using AI and IoT requires a direct hardware bridge to manage standard AC mains-powered appliances. The ESP32 (acting as the IoT hub with AI communication) utilizes its low-power GPIO signals to control a dedicated Relay Driver module. This relay module, typically powered by 5V, provides the necessary electrical isolation and switches the high-voltage (e.g., 230V AC) mains connection. In a common implementation, as pictured below, the ESP32 (powered by 3.3V) and a four-channel Relay Driver (powered by 5V from a common USB source) are connected with their grounds tied together. Signals from the ESP32's GPIO pins go to the Relay Module's inputs, triggering the internal logic to switch the NO (Normally Open) and COM (Common) contacts, thereby activating the load (e.g., an incandescent lamp) connected in series with the AC source. This configuration successfully translates low-level logic commands received via voice into robust power control for household devices.

Fig. 1: Hardware Implementation

SOFTWARE IMPLEMENTATION

Implementing the software for an ESP32-based Automatic Car Parking System involves a main logic loop that manages entry and exit gates based on sensor inputs and slot availability. The system uses two pairs of infrared (IR) sensors: one detects vehicles approaching and clearing the entrance, while the other manages the exit. Upon detecting a car at the entrance, the ESP32 verifies that available Slots
> 0. If space exists, it triggers a PWM signal to open the entry servo gate and decrements the slot count on an I2C LCD. Once the car passes the "clear" sensor, the gate closes. Simultaneously, when a vehicle triggers the exit sensor, the exit gate opens, and the slot count increments upon the car's departure. The ESP32's integrated Wi-Fi can concurrently transmit these real-time occupancy updates to an IoT



platform like Firebase or Blynk for remote monitoring.

Fig. 2: Flowchart

CONCLUSION

The conclusion of the Automatic Car Parking System project demonstrates the successful integration of IoT hardware and automated logic to solve real-world urban challenges. By leveraging the ESP32 microcontroller, the system provides a high-speed, reliable solution for monitoring parking occupancy in real-time. The use of ultrasonic sensors ensures precise vehicle detection, while the I2C LCD and servo-controlled barrier offer a seamless, user-friendly interface for drivers. This project effectively reduces the time spent searching for parking, minimizes traffic congestion within lots, and eliminates the need for manual oversight. Ultimately, this prototype serves as a scalable foundation for "Smart City" infrastructure, proving that low-cost components can be used to build sophisticated, automated systems that enhance operational efficiency and user convenience.

REFERENCE

1. Academic & Journal References

These are useful for citing the "Smart City" and "IoT" aspects of your project.

• MDPI Electronics (2020): *"IoT Based Smart Parking System Using Deep Long Short Memory Network."* This paper discusses the importance of real-time slot detection in reducing urban traffic congestion.

• ResearchGate (2025/Ongoing): *"Design of Smart Parking System Using Ultrasonic Sensor to Optimize Parking Lots."* Focuses specifically on using the HC-SR04 ultrasonic sensor with the ESP32 for vehicle detection and sensitivity analysis.

• World Journal of Advanced Research (2020): *"IoT-Enabled Parking Space Detection Using Ultrasonic Sensors."* Explains the "Time of Flight" formula: $Distance = \frac{Echo\ Time \times 0.034}{2}$, which is the core logic used in your getDistance() function.

2. Technical Component References

These cite the specific hardware datasheets and library documentation.

• Espressif Systems: *"ESP32 Technical Reference Manual."* The official documentation for the ESP32 microcontroller, detailing GPIO pin capabilities and power management.

• LiquidCrystal_I2C Library: Documentation for the Wire.h and LiquidCrystal_I2C.h libraries used to control the 16x2 LCD via the I^2C protocol.

• ESP32Servo Library: Documentation for the ESP32Servo.h library, which explains how the ESP32 handles PWM (Pulse Width Modulation) timers to control the 180° servo motor.

3. Online Project Repositories

Useful for citing existing open-source implementations.

1.GitHub / prince61299: *"Smart Car Parking using ESP32."* A popular repository that