Report

For

Smart Car Parking System

Under the guidance of
Dr. Prateek Raj Gautam

Prepared by

Specialization	SAP ID	Name
MCA - AI/ML	590018354	Sakshi Chauhan

AI Cluster
School Of Computer Science
UNIVERSITY OF PETROLEUM & ENERGY STUDIES,
DEHRADUN- 248007. Uttarakhand

Table of Contents

1. INTRODUCTION

1.1. Purpose of the Project

The Smart Parking System is designed to resolve the common challenge faced by urban drivers: finding an available parking space efficiently. The project aims to use IoT-based technology to automate parking management by detecting vacant spots and updating their status in real-time. This reduces time wastage, fuel consumption, and traffic congestion, improving the overall parking experience.

1.2. Target Beneficiary

The system primarily benefits city drivers, parking facility operators, and urban management authorities. It is suitable for implementation in malls, airports, offices, universities, and public parking areas where congestion and time delays are frequent.

1.3. Project Scope

The Smart Parking System leverages sensors, microcontrollers, and display interfaces to provide a fully automated parking solution. The scope includes real-time monitoring of parking slots, automation, and a mobile/web-based interface for users. The deliverables include the complete hardware setup, an Arduino-based control system, a live data display on LEDs, and a connected web interface.

1.4. References

1.4.1. Al-Kharusi, H., & Al-Bahadili, H. (2019). A Survey on Sensor-Based and Vision-Based Smart Parking Systems. Journal of Information and Knowledge Management.

1.4.2. Bong, D. B. L., Ting, K. H., & Lai, K. C. (2017, September). A review of the smart parking system. In the 2017 International Conference on Engineering Technology and Technopreneurship (ICE2T).

1.4.3. Javed, M. A., Zafar, M. H., & Zafar, S. (2017, December). A review of IoT based smart car parking system. In 2017 International Conference on Innovations in Electrical Engineering and Computational Technologies (ICIEECT).

1.4.4. Parmar, J., Dabhi, J., & Patel, C. (2018). Arduino based smart parking system. International Journal of Trend in Scientific Research and Development (IJTSRD).

2. PROJECT DESCRIPTION

2.1. Reference Algorithm

2.1.1.1. The Smart Parking System using IoT is designed to monitor and display real-time parking slot availability using sensor data and an IoT dashboard. The system uses an Arduino microcontroller, two proximity sensors (such as Ultrasonic or IR sensors), and an OLED display to detect vehicle presence and present live status information. The Arduino collects data from each sensor, updates the OLED display locally, and sends the occupancy status to an IoT platform, where users can view the slot availability on a dashboard.

2.1.1.2. This system aims to reduce congestion, improve parking management, and enhance user convenience by providing accurate, real-time parking information.

2.1.2. System Components

2.1.2.1. Arduino (e.g., UNO/Nano): Serves as the main controller.

2.1.2.2. Two Sensors (IR/Ultrasonic): Detect whether each parking slot is occupied.

2.1.2.3. OLED Display: Shows real-time status of Slot 1 and Slot 2.

2.1.2.4. IoT Platform / Dashboard: Displays live data sent from Arduino for remote monitoring.

2.1.2.5. Connectivity Module (e.g., ESP8266/NodeMCU): Handles cloud communication.

2.1.3. Reference Algorithm

2.1.3.1. Step-by-Step Algorithm for Smart Parking System

2.1.3.2. Start the system

2.1.3.3. Initialize Arduino, sensors, OLED display, and IoT communication module.

2.1.3.4. Initialize all variables

2.1.3.5. Set sensor pins as input.

2.1.3.6. Set OLED display communication.

2.1.3.7. Read sensor data

2.1.3.8. Read value from Sensor 1.

2.1.3.9. Read value from Sensor 2.

2.1.3.10. Determine parking status

2.1.3.11. For each sensor:

a. If measured distance < threshold → Slot Occupied

b. Else → Slot Available

c. Update OLED display

d. Display “Slot 1: Available/Occupied”

e. Display “Slot 2: Available/Occupied”

2.1.3.12. Package data for IoT dashboard

2.1.3.13. Create a payload containing current slot statuses.

2.1.3.14. Send data to the cloud

2.1.3.15. Transmit payload

2.1.3.16. Update dashboard with the latest status.

2.1.3.17. End (only if powered off).

2.2. Characteristic of Data

The Smart Parking System primarily works with real-time sensor data collected from multiple infrared (IR) sensors installed at each parking slot.

2.2.1. Primary Data Source

2.2.1.1. Infrared Sensors (IR):

Generate live, indicating whether a parking slot is occupied or vacant. These readings are continuously collected by the Arduino microcontroller, which updates the parking status in real-time.

2.2.2. Secondary Data Source

2.2.2.1. Web Interface Logs:

Records the parking availability data transmitted from the Arduino to the web dashboard. This serves as a secondary verification dataset for monitoring system accuracy and detecting anomalies such as sensor errors or false readings.

2.3. SWOT Analysis

2.3.1. Strengths

2.3.1.1. Real-time monitoring: Provides instant updates on parking slot availability using IoT sensors.

2.3.1.2. Low-cost implementation: Utilises affordable components like Arduino boards and IR sensors.

2.3.1.3. Scalability: Can be easily expanded to larger parking areas by adding more sensors and controllers.

2.3.1.4. User convenience: Saves time and reduces stress for drivers by displaying available slots before entry.

2.3.1.5. Environmental benefit: Reduces fuel wastage and carbon emissions caused by vehicles searching for parking.

2.3.2. Weaknesses

2.3.2.1. Sensor sensitivity: IR sensors may give inaccurate readings due to dust, rain, or strong sunlight.

2.3.2.2. Limited range: Suitable mainly for small- to medium-sized parking lots without additional networking modules.

2.3.3. Opportunities

2.3.3.1. Mobile app enhancement: Future versions can offer reservation features, digital payment, or navigation to available spots.

2.3.3.2. Commercial adoption: Malls, offices, airports, and institutions can adopt the system for efficient parking management.

2.3.4. Threats

2.3.4.1. Hardware failure: Sensor malfunctions or connectivity loss can impact reliability.

2.3.4.2. Environmental conditions: Harsh weather may affect hardware durability and performance.

2.4. Project Features

The Smart Parking System is designed to automate parking space management through real-time detection, data processing, and display.

Key Features:

2.4.1. Real-Time Parking Slot Detection

2.4.1.1. The system continuously monitors each parking slot using sensors and instantly detects whether a vehicle is present or not.

2.4.2. IoT-Based Live Dashboard

2.4.2.1. Parking slot status is uploaded to an IoT platform, allowing users or administrators to view real-time availability from any location through the dashboard.

2.4.3. OLED Display for Local Status

2.4.3.1. An OLED display installed at the parking area shows the live status of each slot (Occupied/Available), helping users quickly identify vacant spots.

2.4.4. Dual-Sensor Setup for Multiple Slots

2.4.4.1. Two sensors are used to independently monitor two parking slots, enabling scalable and modular expansion for more slots.

2.4.5. Automatic Data Transmission

2.4.5.1. The Arduino sends occupancy data to the cloud automatically at regular intervals using an IoT communication module (e.g., ESP8266/NodeMCU).

2.4.6. Low Power and Cost-Effective Design

2.4.6.1. The system uses low-power components and is cost-efficient, making it suitable for small parking areas, colleges, offices, and residential complexes.

2.4.7. User-Friendly Interface

2.4.7.1. The dashboard provides a clean and easy-to-understand interface showing slot status in real-time, enabling quick decision-making for drivers and administrators.

2.4.8. Reduced Traffic Congestion

2.4.8.1. By providing accurate availability information, the system helps reduce the time spent searching for parking, leading to smoother traffic flow.

2.4.9. Automatic Refresh and Continuous Monitoring

2.4.9.1. The system runs continuously and updates the dashboard and OLED display without manual intervention.

2.4.10. Scalable Architecture

2.4.10.1. Additional sensors and slots can be added to expand the system for larger parking lots without changing the core architecture.

3. SYSTEM REQUIREMENTS

3.1. <u>Hardware Requirements</u>

3.1.1. Microcontroller

3.1.1.1. Arduino Uno / Arduino Nano

3.1.1.2. Handles sensor input processing and OLED display output

3.1.1.3. Sends data to the IoT module

3.1.2. Sensors

3.1.2.1. 2× Ultrasonic Sensors (HC-SR04) or IR Proximity Sensors

3.1.2.2. Used to detect whether a vehicle is present in each parking slot

3.1.3. OLED Display

3.1.3.1. 0.96” I2C OLED Display

3.1.3.2. Displays real-time occupancy status of each slot

3.1.4. IoT Connectivity Module

3.1.4.1. ESP8266 / NodeMCU / ESP32

3.1.4.2. Responsible for connecting to Wi-Fi and updating the IoT dashboard

3.1.5. Power Supply

3.1.5.1. 5V USB or external power supply for Arduino

3.1.5.2. Stable power for sensors and OLED display

3.1.6. Additional Components

3.1.6.1. Jumper wires

3.1.6.2. Breadboard or PCB

3.1.6.3. USB cable for programming

3.2. User Interface:

The Smart Parking System provides two end-user interfaces and one operator interface. Each is real-time and designed for quick decision-making.

A) Entrance LED Display (Driver-facing)

- Purpose: Provide immediate visibility of availability at the .
- Components & Data: Free Slots Counter (large, high-contrast digits)
- Interactions: View-only; updates automatically on each occupancy change.
- Constraints: 16×2 LCD/LED—short text, high contrast, minimal animation.

B) Web Dashboard (Driver & Admin)

- Purpose: Remote, real-time visibility of parking status and basic insights.

3.3. Software Interface

3.3.1. The Smart Parking System includes several software interfaces that connect its hardware components, control logic, and user-facing applications. These interfaces allow real-time communication between sensors, the Arduino microcontroller, motors, the LED display, and the web dashboard.

3.3.2. Arduino Control Interface

3.3.2.1. Purpose: Acts as the system's central logic unit that processes all sensor inputs and controls outputs.

3.3.2.2. Components:

a. IR sensor inputs (digital pins 2–13)

b. LCD outputs

3.3.3. Display Interface

3.3.3.1. Purpose: Provides real-time feedback to drivers through a 16×2 LCD display.

3.3.3.2. Data Flow: Arduino updates display strings whenever FreeCount changes.

3.3.3.3. Example Message: “Available Slots: 05”

3.3.4. Web Dashboard Interface

3.3.4.1. Purpose: Provides a browser-based UI for both drivers and administrators.

3.3.4.2. Technology Stack: HTML / CSS / JavaScript (front end).

3.3.4.3. Endpoints: /updateStatus, /viewStatus, /admin/login.

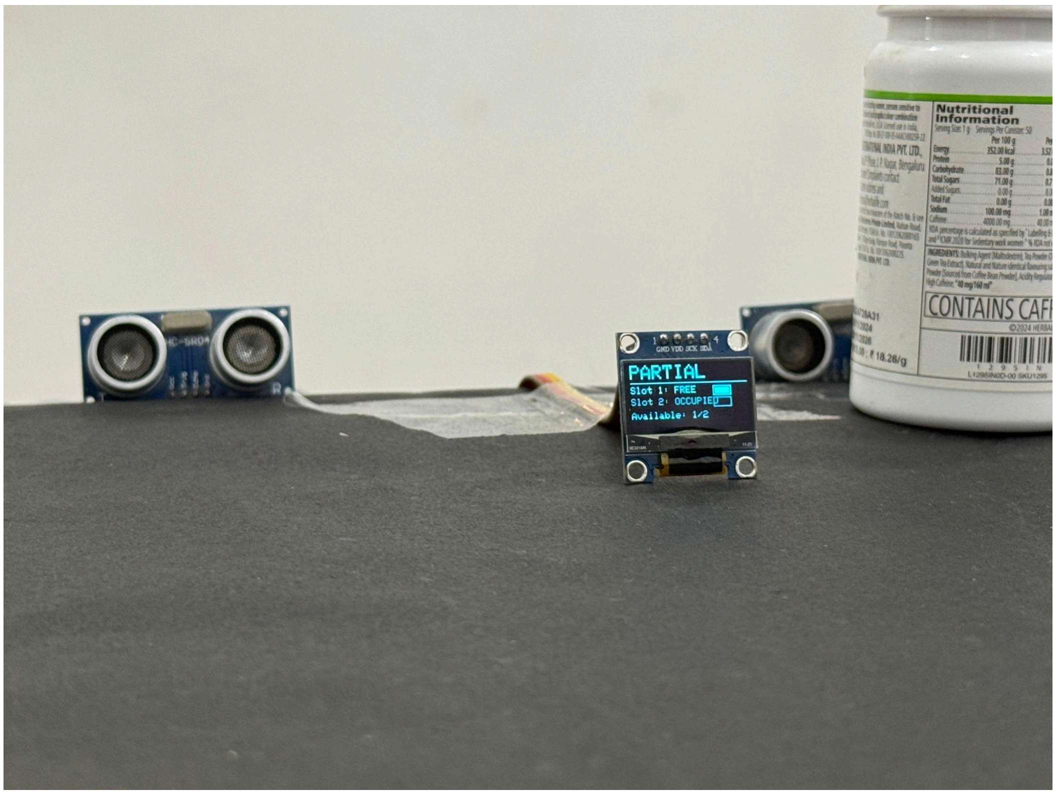
3.4. <u>Database Interface</u>

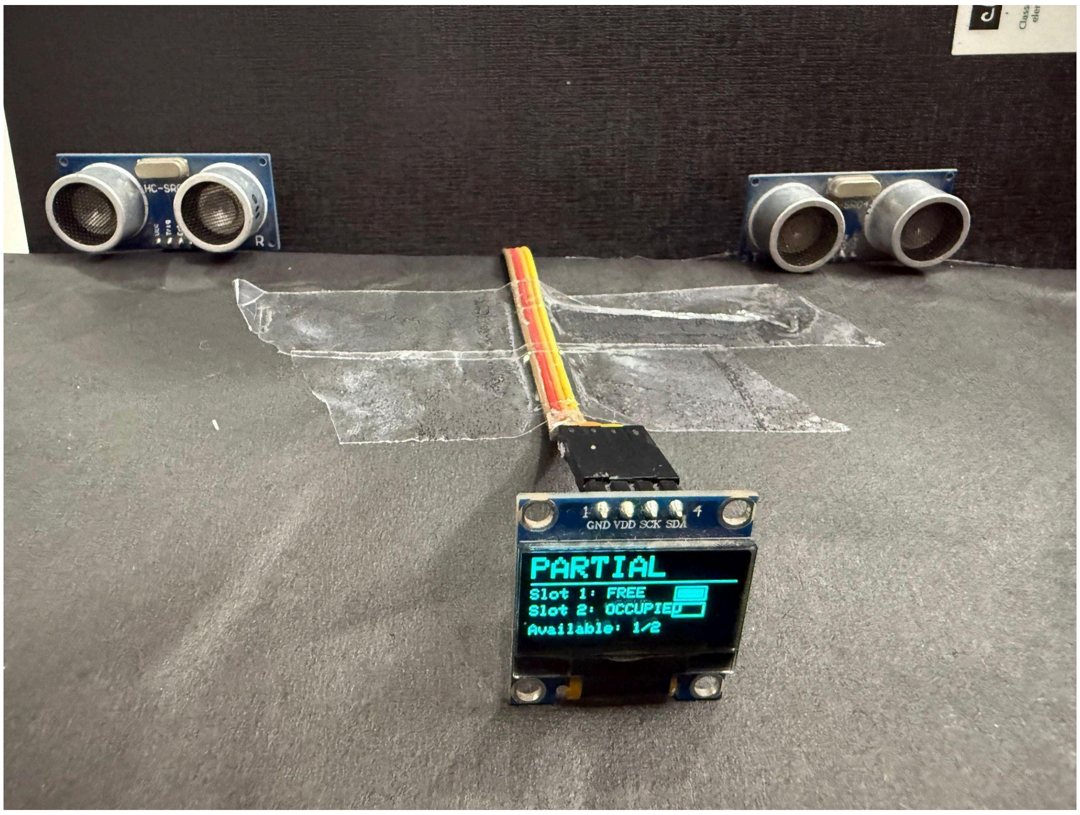
3.4.1. Purpose: Stores historical parking data and system logs for analysis.

3.4.2. Type: Relational (MySQL) or lightweight (NoSQL – Firebase).

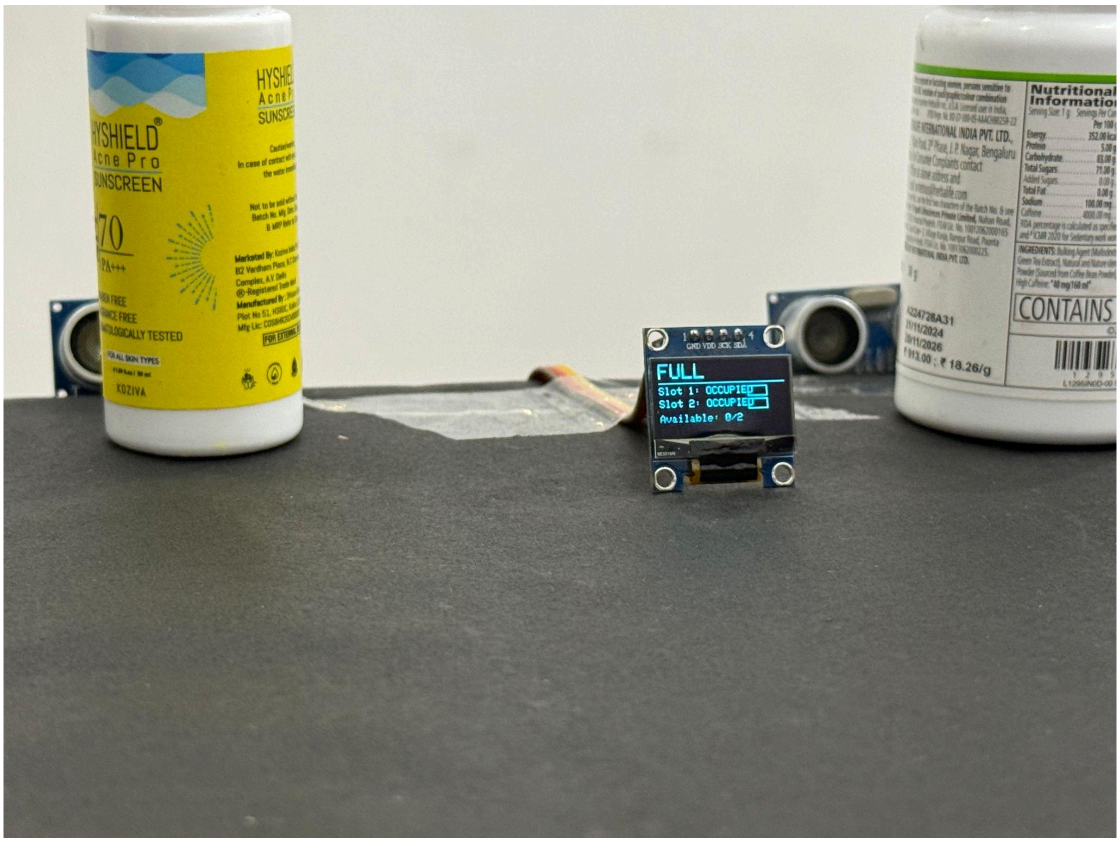
3.4.3. Tables/Collections: slots, transactions, users, errors.

4. EXECUTION

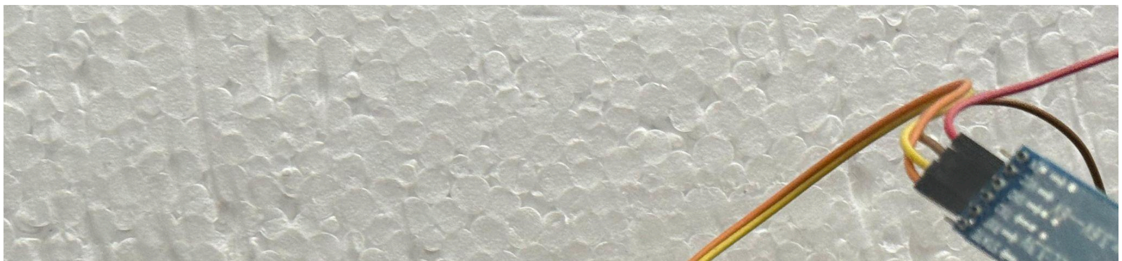

```
PARTIAL
Slot 1: FREE
Slot 2: OCCUPIED
Available: 1/2
```

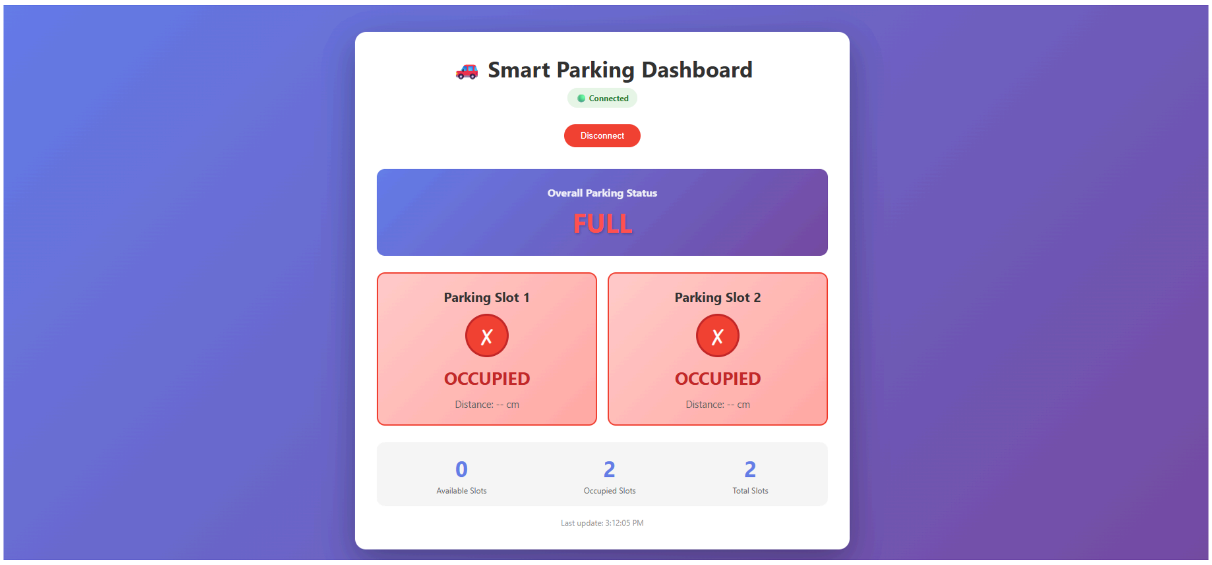


```
PARTIAL
Slot 1: FREE
Slot 2: OCCUPIED
Available: 1/2
```



```
FULL
Slot 1: OCCUPIED
Slot 2: OCCUPIED
Available: 0/2
```




🚗 Smart Parking Dashboard

● Connected

Disconnect

Overall Parking Status

FULL

Parking Slot 1

✗

OCCUPIED

Distance: -- cm

Parking Slot 2

✗

OCCUPIED

Distance: -- cm

0 Available Slots

2 Occupied Slots

2 Total Slots

Last update: 3:12:05 PM

5. NON-FUNCTIONAL REQUIREMENTS

5.1. Performance Requirements: System must update slot status within 2 seconds of change detection.

5.1.1. Real-Time Processing

5.1.1.1. The Arduino microcontroller must process IR sensor inputs and update parking slot status within 1–2 seconds of a vehicle's movement.

5.1.1.2. The LED display and web interface should reflect the latest status update within 2 seconds.

5.1.2. Throughput and Response Time

5.1.2.1. The system must support simultaneous detection of up to 50 sensors without lag.

5.1.2.2. Network transmission delay for data to the web dashboard should not exceed 3 seconds under normal Wi-Fi conditions.

5.1.3. Reliability and Fault Tolerance

5.1.3.1. If a sensor fails, the system must continue functioning and mark that slot as "Sensor Error" on the admin panel.

5.1.3.2. The Arduino should automatically restart in case of a software freeze (using watchdog timer).

5.1.4. Data Update Frequency

5.1.4.1. IR sensor data should be polled at a refresh rate of 1 second.

5.1.4.2. Web dashboard synchronisation interval: every 3–5 seconds.

5.2. Security Requirements: Access to the admin interface secured with login. Data is encrypted during transmission.

5.3. Software Quality Attributes: The System ensures adaptability, reliability, and maintainability. Designed for modular hardware expansion and portable deployment.

5.3.1. Adaptability: The system can be scaled easily by adding more sensors and expanding software parameters for larger parking lots.

5.3.2. Availability: System uptime should be at least 99%, ensuring continuous operation except during scheduled maintenance.

5.3.3. Correctness: Each IR sensor input must reflect the actual state of its corresponding slot, verified through software validation.

5.3.4. Flexibility: The architecture supports integration with future modules (mobile apps, payment systems, analytics dashboards).

5.3.5. Maintainability: Modular code structure allows easy debugging, updates, and component replacement without affecting the entire system.

5.3.6. Reliability: The system ensures stable operation under varying traffic and environmental conditions, using error-checking and watchdog recovery.

5.3.7. Reusability: Individual modules (sensor, control, display) can be reused in other IoT automation projects.

5.3.8. Usability: The interface (LED and web dashboard) is intuitive, with clear messages and minimal user interaction needed.