EPD REPORT

On

SMART PARKING SYSTEM

SUBMITTED BY

CHIRANTAN NANDI (2330228)
GIRISH PATTNAIK (2330230)
SHUBBAN SUNDAR (2330268)
MANIT KUMAR SAHU (2430605)

KALINGA INSTITUTE OF
INDUSTRIAL TECHNOLOGY (KIIT)

Deemed to be University U/S 3 of UGC Act, 1956

B. Tech in Electronics and Computer Science Engineering

School of Electronics Engineering
Kalinga Institute of Industrial Technology Deemed to be University
Bhubaneswar, India

2025

INDEX

INTRODUCTION

With the rapid increase in the number of vehicles in urban areas, parking has become a critical issue. Drivers often waste significant time searching for available parking spaces, which leads to fuel wastage, traffic congestion, and frustration. A **Smart Parking System** aims to solve this issue through **automation and IoT (Internet of Things)** technologies.

In this project, we propose a **Smart Parking System using ESP8266**, where parking slot availability is automatically detected using **Light Dependent Resistors (LDRs)**. The system is designed for **two parking slots**, and its main objective is to display whether each slot is **empty or full** on a **local webpage**. The entire system works through an **Access Point (AP) IP mode**, which means users can connect to the ESP8266 Wi-Fi network **without internet** and still access the live parking status.

This system not only provides convenience to users but also contributes to **smart city development** by promoting digital parking management and reducing vehicle idle time.

OBJECTIVE

1. To design an IoT-based parking management system for monitoring real-time slot availability.

2. To implement an **offline (AP mode)** web-based interface that shows slot status.

3. To use **LDR sensors** for detecting the presence or absence of vehicles.

4. To provide **LED indicators** for quick visual feedback on each slot.

5. To develop a **cost-effective**, **energy-efficient**, and **scalable** parking solution.

COMPONENTS USED

Components	Description / Function
ESP8266 (NodeMCU)	Acts as the central controller and Wi-Fi access point. Hosts the local webpage displaying slot status.
LDR Sensor (2 units)	Detects light intensity. Used to determine if a vehicle is parked over the slot
Resistors	Used in voltage divider circuits for LDRs to generate measurable analog signals.
LEDs (Red & Green)	Indicate slot status: Green for Empty, Red for Full.
Breadboard & Jumper Wires	For circuit assembly and testing.
5V Power Supply / USB Cable	Provides power to the ESP8266 and sensors.

WORKING PRINCIPLE

Each parking slot is equipped with an LDR sensor. The working of the system can be explained in the following steps:

1. **VehicleDetection:**

   When no vehicle is present, the LDR receives normal ambient light. When a vehicle parks over the slot, the light is blocked, causing the resistance of the LDR to increase.

2. **SignalProcessing:**

   The ESP8266 continuously reads the analog values from the LDRs. Based on a predefined threshold, it determines whether a slot is "Empty" or "Full".

3. **Indication:**

   - Green LED glows when the slot is empty.
   - Red LED glows when the slot is full.

4. **WebpageDisplay:**

   The ESP8266 works in Access Point mode, creating a local Wi-Fi network.

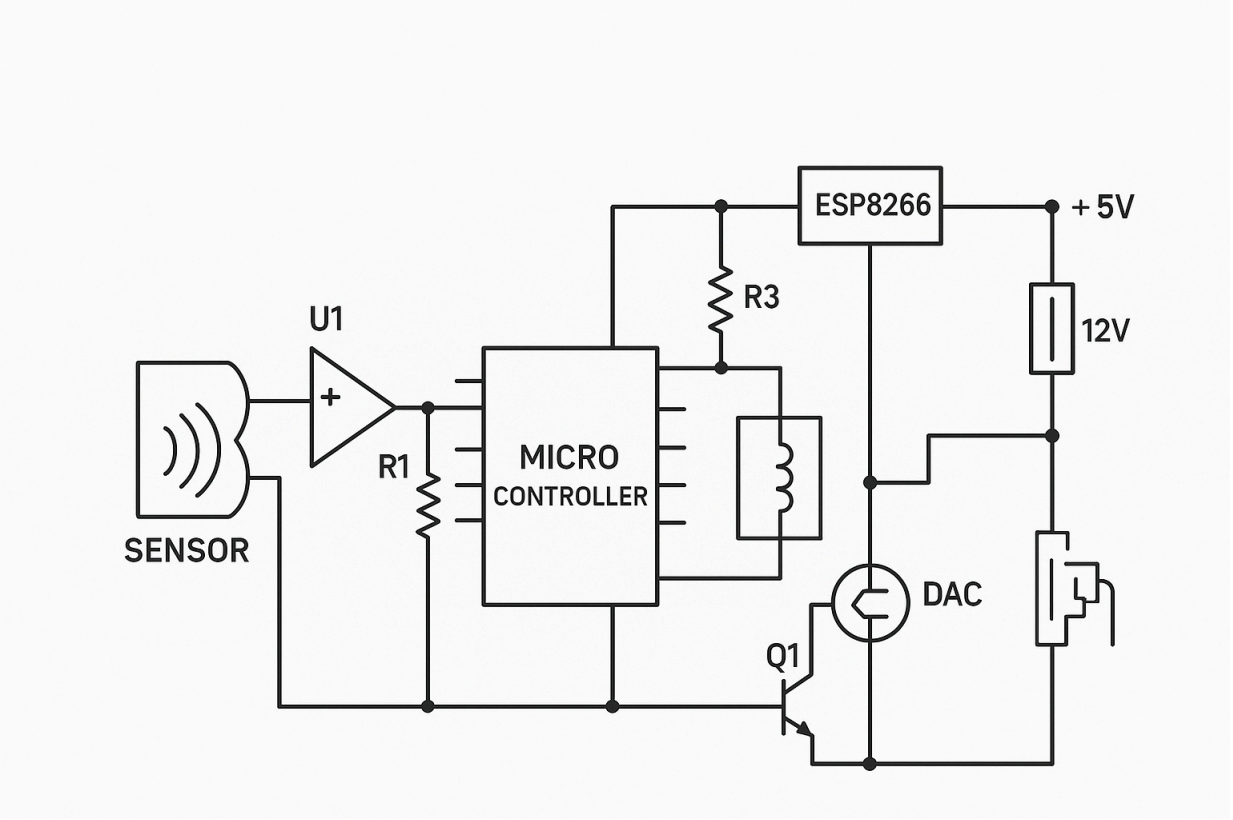
   Users can connect their smartphones or laptops to this network (for example, SSID: *SmartParking_ESP8266*).

   When they open the browser and enter the Access Point IP (e.g., 192.168.4.1), they will see a dashboard webpage showing:

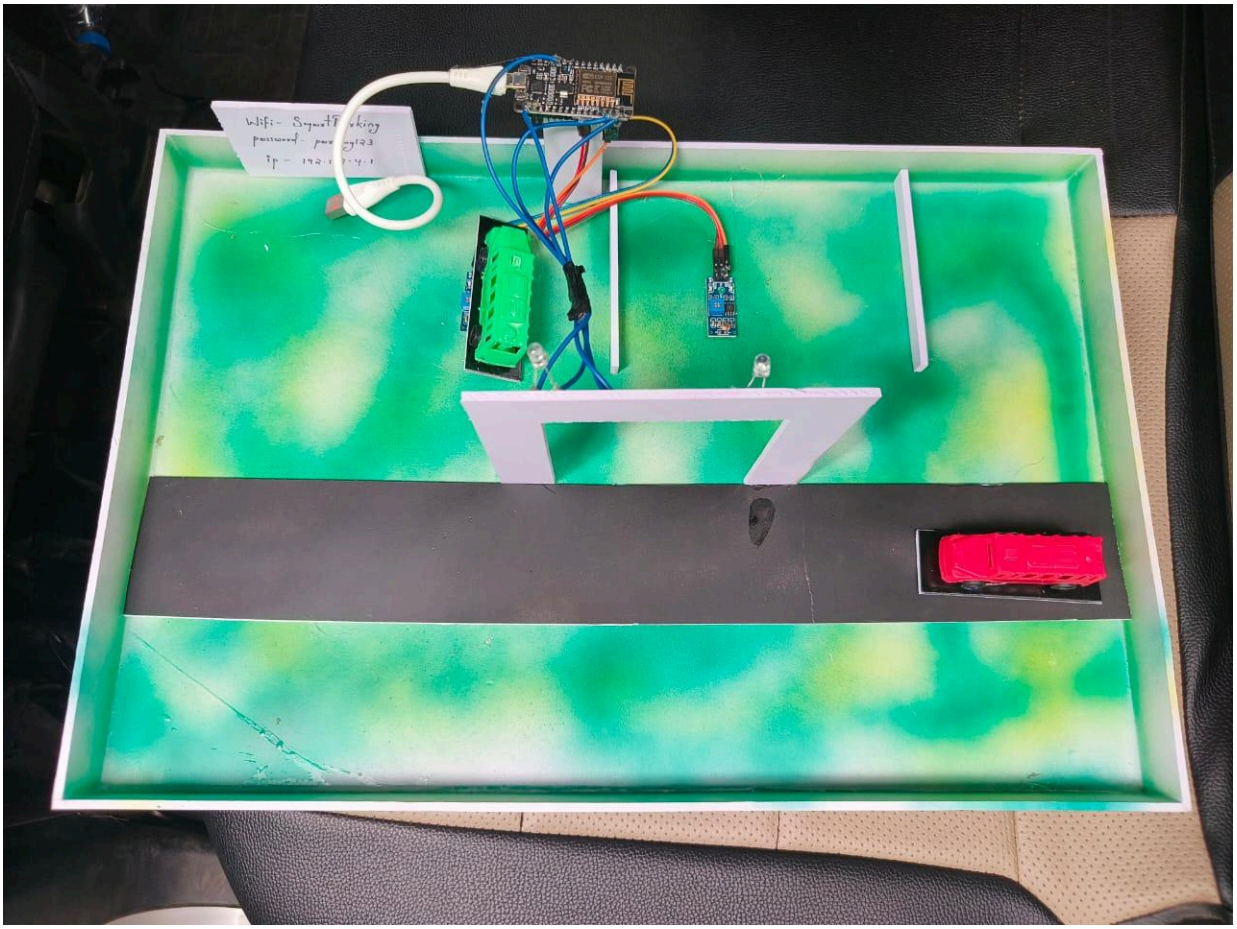
   - Slot 1: Full / Empty
   - Slot 2: Full / Empty

The webpage updates in real time according to the sensor readings.

CIRCUIT DIAGRAM



SMART PARKING SYSTEM



ADVANTAGES

1. Real-time Monitoring: Instantly displays slot availability.
2. Offline Operation: Works without internet using AP IP mode.
3. User-Friendly Interface: Simple and attractive web dashboard.
4. Energy Efficient: Consumes very little power.
5. Scalable Design: Can easily be extended to multiple slots or larger parking areas.

APPLICATIONS

1. Shopping malls and multiplexes
2. Office and residential complexes
3. Hospitals and railway stations
4. Smart city and campus parking lots

FUTURE SCOPE

1. Integration with mobile applications and cloud databases for large-scale parking management.
2. Use of ultrasonic sensors for more accurate detection.
3. Automatic gate control when a vehicle enters or leaves.
4. Integration with payment systems for digital parking fees.
5. Addition of solar panels to make the system self-sustainable.
6. AI-based analytics for parking pattern analysis and optimization.

CONCLUSION

The Smart Parking System using ESP8266 provides a practical and efficient solution for parking management in modern cities. By combining IoT technology, sensor-based detection, and AP IP web visualization, it reduces human effort, traffic congestion, and environmental impact.

The project demonstrates how simple hardware components and intelligent coding can create a reliable and user-friendly smart system. It serves as a foundation for future smart infrastructure developments and supports the concept of sustainable smart cities.

CODE

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>


const char* ssid = "SmartParking";
const char* password = "parking123";

ESP8266WebServer server(80);
const int irSensors[2] = {d1, d2};
bool parkingStatus[2] = {false, false};

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML>
<html>
<head>
  <title>Smart Parking System</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <style>
    body { font-family: Arial, sans-serif; text-align: center; }
    .parking-spot {
      display: inline-block;
      width: 150px;
      height: 100px;
      margin: 10px;
      line-height: 100px;
      font-size: 18px;
      color: white;
    }
    .free { background-color: green; }
    .occupied { background-color: red; }
  </style>
</head>
<body>
  <h1>Smart Parking System</h1>
  <div id="parking-spots"></div>
  <script>
    function updateStatus() {
      fetch('/status')
        .then(response => response.json())
        .then(data => {
          const spotsDiv = document.getElementById('parking-spots');
          spotsDiv.innerHTML = '';
          data.forEach((status, index) => {
            const spot = document.createElement('div');
            spot.className = parking-spot ${status ? 'occupied' : 'free'};
            spot.textContent = Parking ${index + 1} ${status ? 'Full' : 'is Free'};
            spotsDiv.appendChild(spot);
          });
        });
    }
    setInterval(updateStatus, 1000);
    updateStatus();
  </script>
</body>
</html>
)rawliteral";

void setup() {
  Serial.begin(115200);


  server.on("/", handleRoot);
  server.on("/status", handleStatus);
  server.begin();
  Serial.println("HTTP server started");
}

void loop() {
  server.handleClient();
  updateParkingStatus();
}

void updateParkingStatus() {
  for (int i = 0; i < 2; i++) {
    parkingStatus[i] = digitalRead(irSensors[i]);
  }
}

void handleRoot() {
  server.send(200, "text/html", index_html);
}

void handleStatus() {
  DynamicJsonDocument doc(100);  // Smaller size since we only have 2 slots


  for (int i = 0; i < 2; i++) {
    array.add(parkingStatus[i]);
  }

  String response;
  serializeJson(doc, response);
  server.send(200, "application/json", response);
}
```