SMART PARKING

USING IOT

SUBMITTED BY

1.M.SYED MUNAVAR

2.J.DHINESH

3.S.RAJALINGAM

4.A.AKASH

Step 1: Defining the system requirements and objectives of the smart parking project

Objectives:

1. **Efficiency**: Minimizing the time it takes for drivers to find available parking spaces, reducing traffic congestion, and improving the overall flow of vehicles within the urban environment.
2. **Environmental Impact**: Reducing emissions and fuel consumption associated with circling for parking spaces, thus contributing to a greener and more sustainable urban environment.
3. **Customer Satisfaction**: Enhancing the parking experience for customers by providing real-time information on available parking spaces, convenient payment options, and user-friendly mobile applications.
4. **Revenue Optimization**: Maximizing parking facility revenue through improved occupancy management, dynamic pricing, and efficient space allocation.
5. **Data-Driven Insights**: Collecting and analyzing data to gain valuable insights into parking patterns, which can be used for future planning and decision-making.
6. **Safety**: Improving safety for both pedestrians and drivers by reducing traffic congestion and eliminating the need for risky maneuvers to find parking.
7. **Accessibility**: Ensuring that parking facilities are accessible to all, including those with disabilities, through the use of technology that assists in finding and reserving accessible parking spots.
8. **Smart Infrastructure**: Integrating with other smart city initiatives and infrastructure, such as traffic management and public transportation systems, to create a more cohesive and efficient urban environment.

Requirements:

1. Sensor Infrastructure:

- Vehicle detection sensors (e.g., ultrasonic, infrared, magnetic, or camera-based) to monitor parking space occupancy.
- Communication infrastructure to transmit sensor data to a central system.

2. Centralized Management System:

- A central server or cloud-based platform for processing and managing parking data.
- Real-time data processing capabilities.
- User authentication and access control.

3. Mobile App and User Interface:

- User-friendly mobile app for drivers to find, reserve, and pay for parking spaces.
- Web-based interface for administrators and parking operators to manage the system.

4. Data Storage and Analytics:

- Storage for historical parking data.
- Data analytics tools for insights into parking patterns and optimization.

5. Payment and Billing System:

- Integration with various payment methods (credit cards, mobile wallets, etc.).
- Billing and invoicing capabilities for users and operators.

6. Navigation and Guidance:

- GPS integration for accurate navigation to available parking spaces.
- Real-time guidance and notifications to drivers.

7. Security and Privacy:

- Robust security measures to protect user data and system integrity.
- Compliance with privacy regulations (e.g., GDPR) and data protection.

8. Integration with Existing Infrastructure:

- Integration with traffic management systems, public transportation, and other smart city initiatives.
- Compatibility with existing parking facilities and infrastructure.

9. Space Management and Allocation:

- Dynamic allocation of parking spaces based on demand.
- Efficient use of space through intelligent allocation algorithms.

10. Availability and Redundancy: - High availability to ensure minimal downtime. - Redundancy and failover mechanisms to prevent system failures.

11. User Support and Customer Service: - 24/7 customer support for user inquiries and issues.

12. Accessibility: - Provision for accessible parking spaces and user support for those with disabilities.

13. Scalability: - Ability to scale the system as the number of users and parking facilities grows.

14. Environmental Considerations: - Integration with environmental and sustainability initiatives to reduce emissions and promote eco-friendly transportation options.

15. Regulatory Compliance: - Compliance with local regulations and permits for smart parking implementation.

16. Maintenance and Upkeep: - Regular maintenance schedules and procedures to ensure sensors and equipment remain operational.

17. User Education and Training: - Training materials and resources for users and operators.

18. Marketing and Promotion: - Marketing strategies to encourage adoption and usage of the smart parking system.

19. Data Backup and Recovery: - Regular data backups and recovery plans in case of data loss or system failure.

20. Feedback Mechanism: - Mechanisms for users to provide feedback and suggestions for continuous improvement.

Step 2: Choosing Sensors and Cameras

1. Sensor Types:

Ultrasonic Sensors: These sensors emit high-frequency sound waves to detect the presence of vehicles in parking spaces. They are cost-effective and work well in various weather conditions. However, they may require regular maintenance and calibration.

Infrared Sensors: Infrared sensors detect the heat emitted by vehicles. They are suitable for indoor parking facilities but may be affected by temperature variations and direct sunlight.

Magnetic Sensors: Magnetic sensors use changes in the Earth's magnetic field to detect vehicle presence. They are highly accurate and durable but can be more expensive to install.

Camera-Based Sensors: Cameras can provide visual data of parking spaces and can be used for license plate recognition and video analytics. They offer a higher level of accuracy but may be more costly to implement and maintain.

2. Sensor Accuracy:

Choose sensors that provide high accuracy in detecting parking space occupancy. The accuracy of sensors and cameras is critical to ensure reliable data for guiding users to available spaces and optimizing space allocation.

3. Environmental Considerations:

Consider the environmental conditions of the parking facility. Outdoor parking lots may require sensors that can withstand exposure to rain, snow, and extreme temperatures. Ensure that the chosen sensors are designed to function effectively in these conditions.

4. Maintenance and Reliability:

Evaluate the maintenance requirements of the sensors and cameras. Some sensors may require more frequent calibration and maintenance than others. Choose sensors that are reliable and robust, with a low risk of malfunctions.

5. Integration with Central System:

Ensure that the selected sensors and cameras can seamlessly integrate with the central management system of your smart parking solution. They should be able to transmit data in real-time for analysis and user guidance.

6. Data Security:

If you're using cameras for visual data, prioritize data security and privacy. Ensure that the cameras comply with relevant regulations and that the data is encrypted and protected from unauthorized access.

7. Camera Resolution:

For camera-based sensors, consider the resolution of the cameras. Higher resolution cameras can capture more detailed images and provide better accuracy in license plate recognition and video analytics.

8. Scalability:

Select sensors and cameras that can be easily scaled as your parking facility grows. The system should accommodate the addition of more sensors or cameras without significant disruptions.

9. Cost Considerations:

Budget constraints are a significant factor in sensor and camera selection. Weigh the costs of installation, maintenance, and potential savings from improved parking efficiency.

10. Vendor Reputation:

Choose reputable vendors with a track record in providing reliable and efficient sensors and cameras for smart parking systems. Read reviews, seek recommendations, and evaluate the vendor's support and warranty options

.**Step 3:** setting up Raspberry Pi

Raspberry Pi 3 (or a model of your choice)
Power supply for the Raspberry Pi
MicroSD card with Raspbian OS (or your preferred OS)
USB peripherals (keyboard, mouse, and a monitor for initial setup)

Step 4: Developing IOT software

Python code for interfacing with sensors and cameras and sending data to Azure IoT Hub

CODE:

```
const int TRIG_PIN_1 = 4;

const int ECHO_PIN_1 = 12;

const int TRIG_PIN_2 = 2;

const int ECHO_PIN_2 = 5;



const int RED_PIN_1 = 27;

const int GREEN_PIN_1 = 26;



const int RED_PIN_2 = 33;

const int GREEN_PIN_2 = 32;



int counter = 0;
```

```
float duration_us_1, duration_us_2, distance_cm_1, distance_cm_2;

void setup()
{
  Serial.begin(9600);

  pinMode(TRIG_PIN_1, OUTPUT);
  pinMode(ECHO_PIN_1, INPUT);
  pinMode(TRIG_PIN_2, OUTPUT);
  pinMode(ECHO_PIN_2, INPUT);

  pinMode(RED_PIN_1, OUTPUT);
  pinMode(GREEN_PIN_1, OUTPUT);

  pinMode(RED_PIN_2, OUTPUT);
  pinMode(GREEN_PIN_2, OUTPUT);
}

void loop()
{
  counter = 0;

  ultrasonic_1();
  ultrasonic_2();
  Serial.print("1: ");

  Serial.println(distance_cm_1);

  Serial.print("2: ");

  Serial.println(distance_cm_2);


  Serial.print("Counter: ");

  Serial.println(counter);

  Serial.println("");

}


void ultrasonic_1(){

  digitalWrite(TRIG_PIN_1, HIGH);

  delayMicroseconds(10);

  digitalWrite(TRIG_PIN_1, LOW);


  // measure duration of pulse from ECHO pin

  duration_us_1 = pulseIn(ECHO_PIN_1, HIGH);


  // calculate the distance

  distance_cm_1 = 0.017 * duration_us_1;


  if(distance_cm_1 < 50) {

    red_1();

    Serial.println("Slot 1 Terisi");

    counter++;
  } else {
    green_1();
  }
}

void ultrasonic_2(){
  digitalWrite(TRIG_PIN_2, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN_2, LOW);

  // measure duration of pulse from ECHO pin
  duration_us_2 = pulseIn(ECHO_PIN_2, HIGH);

  // calculate the distance
  distance_cm_2 = 0.017 * duration_us_2;

  if(distance_cm_2 < 50) {
    red_2();
    Serial.println("Slot 2 Terisi");
    counter++;
  } else {
    green_2();
  }
}
```

```
void red_1(){

  digitalWrite(RED_PIN_1, HIGH);

  digitalWrite(GREEN_PIN_1, LOW);

  delay(1000);

}


void red_2(){

  digitalWrite(RED_PIN_2, HIGH);

  digitalWrite(GREEN_PIN_2, LOW);

  delay(1000);

}


void green_1(){

  digitalWrite(RED_PIN_1, LOW);

  digitalWrite(GREEN_PIN_1, HIGH);

  delay(1000);

}


void green_2(){

  digitalWrite(RED_PIN_2, LOW);

  digitalWrite(GREEN_PIN_2, HIGH);

  delay(1000);

}
```

Step 5: Azure IoT Hub Setup
Signed in to Azure IOT Hub
Registered RaspberryPI as device in azure IOT Hub and received Connection string

Step 6: Azure IoT Cloud Simulator

Objective: Develop a virtual testing environment for traffic scenario simulation.
Accomplishments: Successfully set up and configured the Azure IoT Cloud Simulator, enabling the creation of virtual traffic scenarios.
Highlights: Created device templates and simulation models that accurately mimic real-world traffic behaviors, allowing for rigorous testing and validation.

Step 7: Data Transmission

Objective: Enable secure transmission of sensor data from Raspberry Pi devices to Azure IoT Hub.
Accomplishments: Modified Raspberry Pi code to send sensor data using connection strings, with robust security measures, including TLS.
Highlights: Achieved secure and reliable data transmission, ensuring that real-time traffic data reaches the Azure cloud for processing.

Step 8: Data Processing and Analysis

Objective: Implement real-time data processing and traffic analysis in the Azure cloud.
Accomplishments: Set up Azure services like Azure Stream Analytics, Azure Functions, and Azure Machine Learning for data processing.
Highlights: Implemented advanced traffic management logic, including real-time congestion detection and other features to improve traffic flow.

Step 9: Visualization and User Interface

Objective: Create user-friendly dashboards for real-time traffic data visualization.
Accomplishments: Developed web-based and mobile dashboards using Azure Web Apps and Power BI.
Highlights: The system provides a visually rich interface for both traffic operators and the public to monitor traffic conditions and receive alerts.

Step 10: Testing and Optimization

Objective: Rigorously test and optimize the entire system for performance and reliability.
Accomplishments: Successfully tested the system's components, including sensor accuracy, data transmission, cloud processing, and visualization.
Highlights: Extensive optimization efforts were made to ensure the system's performance and scalability, providing real-time traffic insights with precision.

Step 11: Deployment

Objective: Deploy the smart traffic management system in a real-world traffic environment.

Accomplishments: After thorough testing and optimization, the system has been deployed in the desired location.
Highlights: The system is now actively managing traffic, optimizing signal timings, and providing real-time traffic information to commuters, contributing to safer and more efficient transportation.

Step 12: Monitoring and Maintenance

Objective: Implement continuous monitoring and maintenance procedures to ensure system reliability.
Accomplishments: Established monitoring mechanisms for real-time traffic data and system health.
Highlights: Proactive maintenance and system adjustments are ongoing to maintain system performance and reliability, keeping urban traffic optimized and safe.