|  |
| --- |
| USING IOT |
| SMART PARKING |
| PHASE-5 |

TEAM MEMBERS:

* JEGADEESH.J **( LEADER** )
* LENIN.S
* DHANUSH.P
* AJAYPRANESH.K
* GOVINDASWAMY.K

Smart parking systems that use IoT (Internet of Things)

This technology aim to make parking more efficient, convenient, and environmentally friendly. These systems leverage various sensors, communication networks, and software applications to optimize parking management.

1, **Sensor Deployment**: IoT-based smart parking systems rely on sensors deployed in parking spaces or areas. These sensors can be of different types, including:

a. **Ultrasonic or Infrared Sensors**: These sensors detect the presence of vehicles in individual parking spaces by measuring the distance between the sensor and the vehicle.

b. **Magnetic Sensors**: Magnetic field sensors can detect changes in magnetic fields caused by the presence of a vehicle.

c. **Image Recognition Cameras**: Cameras equipped with image recognition software can capture images of parking spaces and analyze them to determine occupancy.

d. **RFID or NFC Tags**: RFID (Radio-Frequency Identification) or NFC (Near Field Communication) tags can be placed on vehicles to identify and track them as they enter and exit parking areas.

2, Data Collection and Communication: The sensors collect data on parking space occupancy and transmit it to a central control system or a cloud-based platform using wireless communication technologies like Wi-Fi, Bluetooth, or cellular networks. The data can include information about the location, availability, and status of parking spaces.

3, Data Processing and Analysis: The collected data is processed and analyzed in real-time to provide insights and actionable information. Algorithms can determine the availability of parking spaces, calculate occupancy rates, and predict future parking demand.

4, User Interface: Users can access the parking information through various interfaces, including mobile apps, websites, or electronic displays at parking entrances. These interfaces provide real-time data on available parking spaces, pricing, and navigation to the nearest available spot.

5, Reservation and Payment: Some smart parking systems allow users to reserve parking spaces in advance through mobile apps or websites. Payment for parking can also be made electronically through the same platform, reducing the need for physical payment methods like coins or tokens.

6, Navigation Assistance: IoT-based smart parking systems often provide navigation assistance to guide drivers to the nearest available parking spot. This can be particularly useful in large parking facilities.

7, Analytics and Reporting: The system can generate reports and analytics for parking operators, helping them optimize pricing, capacity planning, and overall management of the parking facility.

8, Sustainability and Efficiency: Smart parking systems help reduce traffic congestion and emissions by guiding drivers directly to available parking spaces, reducing the time spent searching for parking.

9, Remote Monitoring and Maintenance: Operators can remotely monitor the status of the sensors and receive alerts for maintenance or repair as needed, ensuring the system's reliability.

10, Scalability: IoT-based smart parking solutions are scalable, allowing additional sensors to be easily added as parking facilities expand.

**Components:**

1. **Raspberry Pi:** You can use a Raspberry Pi (preferably a Raspberry Pi 3 or later) as the central controller.
2. **Ultrasonic Sensors:** Ultrasonic sensors can be used to detect the presence of vehicles in each parking space.
3. **LEDs or Displays:** LEDs or small displays can be used to indicate the status of each parking space (e.g., red for occupied, green for vacant).
4. **Power Supply:** Ensure you have a power source for your Raspberry Pi and sensors.
5. **Internet Connectivity:** A Wi-Fi dongle or an Ethernet connection to connect the Raspberry Pi to the internet.

Block Diagram:

**Steps to Create a Smart Parking System:**

1. **Set up the Raspberry Pi:**

a. Install the Raspberry Pi OS (Raspbian) on your Raspberry Pi.

b. Configure Wi-Fi or Ethernet for internet connectivity.

c. Set up remote access to your Raspberry Pi for ease of management.

1. **Connect Ultrasonic Sensors:**

a. Connect the ultrasonic sensors to the Raspberry Pi GPIO pins. Ultrasonic sensors have four pins - VCC (power), GND (ground), TRIG (trigger), and ECHO (echo).

b. Write and test Python code to interact with the ultrasonic sensors. The code should send a pulse from the TRIG pin and measure the time it takes for the pulse to return to the ECHO pin. This time can be used to calculate the distance to the object (vehicle) in front of the sensor.

1. **LEDs or Displays:**

a. Connect LEDs or small displays to the Raspberry Pi GPIO pins to indicate parking space status. For example, you can use a red LED for an occupied space and a green LED for a vacant space.

b. Write code to update the status of LEDs or displays based on the data received from the ultrasonic sensors.

1. **Data Processing:**

a. Develop Python scripts to process the sensor data and determine parking space occupancy.

b. You can maintain a database or a data structure in memory to keep track of parking space status.

1. **User Interface:**

a. Create a simple web-based interface to display the parking space status. You can use Flask or Django for web development on the Raspberry Pi.

b. Alternatively, you can create a mobile app or a web app that communicates with the Raspberry Pi to display real-time parking space availability.

1. **Remote Access and Monitoring:**

a. Implement remote monitoring capabilities for your system. You can use SSH or a VPN for secure remote access.

b. Set up notifications or alerts (e.g., email or SMS) to notify you in case of issues or to provide updates on parking space status.

1. **Mount and Secure Sensors:**

a. Mount the ultrasonic sensors at each parking space, ensuring they have a clear line of sight to vehicles.

b. Securely attach the Raspberry Pi and any other electronics in a weatherproof enclosure if necessary.

1. **Testing and Calibration:**

a. Test the system by parking vehicles in the spaces and observing how the LEDs or displays respond.

b. Fine-tune the system as needed to improve accuracy and reliability.

1. **Deployment:**

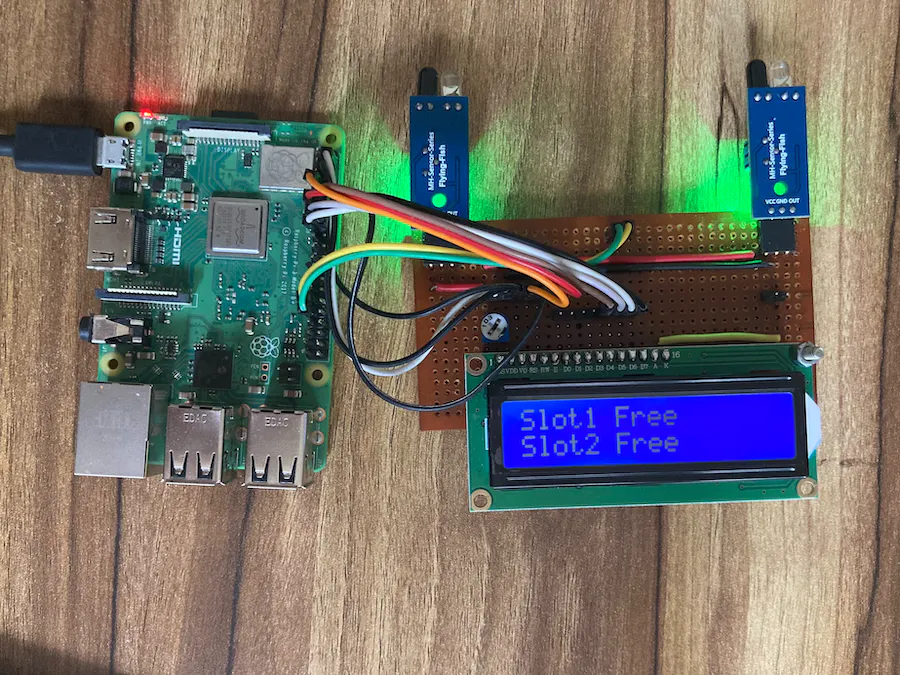
a. Install the system in your parking area, and ensure it has a reliable power source.

b. Make the user interface accessible to the intended users.

1. **Documentation and Maintenance:**

a. Document the system's setup, configurations, and any maintenance procedures.

b. Regularly monitor and maintain the system to ensure it functions as intended.



Python Code:

import RPi.GPIO as GPIO

import time

# Define GPIO pins for the ultrasonic sensor and LED

TRIG\_PIN = 18 # Trigger pin

ECHO\_PIN = 24 # Echo pin

LED\_PIN = 21 # LED pin

# Set up GPIO mode and warnings

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)

# Set up GPIO pins

GPIO.setup(TRIG\_PIN, GPIO.OUT)

GPIO.setup(ECHO\_PIN, GPIO.IN)

GPIO.setup(LED\_PIN, GPIO.OUT)

# Function to measure distance using the ultrasonic sensor

def measure\_distance():

# Trigger ultrasonic sensor

GPIO.output(TRIG\_PIN, True)

time.sleep(0.00001)

GPIO.output(TRIG\_PIN, False)

# Wait for the echo to be high and then low

while GPIO.input(ECHO\_PIN) == 0:

pulse\_start = time.time()

while GPIO.input(ECHO\_PIN) == 1:

pulse\_end = time.time()

# Calculate distance

pulse\_duration = pulse\_end - pulse\_start

distance = pulse\_duration \* 17150 # Speed of sound (343 m/s) divided by 2

return distance

try:

while True:

distance = measure\_distance()

if distance < 10: # Adjust this distance threshold as needed

print("Parking space occupied")

GPIO.output(LED\_PIN, GPIO.HIGH) # Turn on the LED

else:

print("Parking space vacant")

GPIO.output(LED\_PIN, GPIO.LOW) # Turn off the LED

time.sleep(2) # Adjust the interval between distance measurements

except KeyboardInterrupt:

GPIO.cleanup()

In this code:

* We use the RPi.GPIO library to control the GPIO pins on the Raspberry Pi.
* The measure\_distance function calculates the distance from the ultrasonic sensor to an object (vehicle).
* We define a threshold distance (10 centimeters in this example) to determine if the parking space is occupied or vacant.
* An LED connected to the GPIO pin is turned on when the space is occupied and turned off when the space is vacant.

Creating a web interface for a smart parking system

1. **Choose a Web Development Stack:**

Decide on the web development stack you want to use. Common technologies include HTML, CSS, JavaScript for the frontend, and a backend framework such as Flask, Django, Node.js, or Ruby on Rails for handling data and user interactions.

1. **Design the User Interface (UI):**

Plan the layout and design of your web application. Consider the following elements:

* + Parking space status display (e.g., a map or list).
  + Real-time updates to show space availability.
  + User registration and login (if needed).
  + Reservation and payment options (if applicable).
  + Notifications and alerts.
  + Navigation assistance.

1. **Frontend Development:**

Write the frontend code for your web application. Here's a basic example using HTML, CSS, and JavaScript:

* + Create HTML templates for the UI elements.
  + Use CSS to style the interface.
  + Use JavaScript to fetch and display real-time parking space data from your Raspberry Pi-based smart parking system.

1. **Backend Development:**

Depending on your chosen backend framework, set up routes, controllers, and models. In the backend, you'll handle:

* + Data retrieval from your Raspberry Pi or smart parking system.
  + User authentication and authorization (if needed).
  + Data processing and presentation to the frontend.

1. **Database Integration (if needed):**

If your smart parking system requires storing data, set up a database. Common choices include MySQL, PostgreSQL, or SQLite for structured data, or NoSQL databases like MongoDB for flexible data storage.

1. **Real-Time Updates:**

For real-time updates on parking space availability, consider using WebSockets or server-sent events (SSE) to push updates from the backend to the frontend. Libraries like Socket.io (for Node.js) or Django Channels (for Django) can help with real-time communication.

1. **Testing:**

Test your web application thoroughly. Check for any bugs or issues in both the frontend and backend. Ensure that real-time updates are working correctly.

1. **Deployment:**

Choose a hosting platform for your web application. Popular options include AWS, Heroku, DigitalOcean, or a Raspberry Pi web server setup. Configure your server environment, including web server software like Nginx or Apache.

1. **Security:**

Implement security measures to protect your web application, including securing user data, handling user authentication, and preventing common web vulnerabilities like SQL injection and cross-site scripting (XSS).

1. **Monitoring and Maintenance:**

Regularly monitor your web application for performance, scalability, and security. Make necessary updates and improvements based on user feedback and evolving requirements.

1. **Documentation:**

Document your web application, including installation instructions and usage guidelines for users and developers.

1. **User Support and Feedback:**

Provide user support, and consider adding features that enable users to provide feedback or report issues.

1. **Scalability and Future Enhancements:**

Plan for future enhancements and scalability, especially if you intend to expand your smart parking system.

Web Developing Code:

Creating a complete web-based smart parking system involves a significant amount of code and is beyond the scope of a single response. However, I can provide a basic example of how to create a simple web interface to display parking space availability using HTML, CSS, and JavaScript. You can build upon this example and integrate it with your Raspberry Pi-based smart parking system.

Here's a simplified HTML, CSS, and JavaScript code snippet for a basic parking space status display:

<!DOCTYPE html>

<html>

<head>

<title>Smart Parking System</title>

<style>

/\* Add your CSS styles for the interface here \*/

.parking-space {

width: 100px;

height: 100px;

display: inline-block;

text-align: center;

font-size: 20px;

margin: 10px;

padding: 10px;

}

.vacant {

background-color: #4CAF50; /\* Green \*/

color: white;

}

.occupied {

background-color: #F44336; /\* Red \*/

color: white;

}

</style>

</head>

<body>

<h1>Smart Parking System</h1>

<div id="parking-status">

<div class="parking-space vacant">Space 1</div>

<div class="parking-space occupied">Space 2</div>

<div class="parking-space vacant">Space 3</div>

</div>

<script>

// Simulated parking space data; replace with real-time data from your system

const parkingData = [true, false, true]; // true for vacant, false for occupied

// Function to update parking space status

function updateParkingStatus() {

const parkingSpaces = document.querySelectorAll(".parking-space");

parkingData.forEach((status, index) => {

if (status) {

parkingSpaces[index].classList.remove("occupied");

parkingSpaces[index].classList.add("vacant");

parkingSpaces[index].textContent = `Space ${index + 1} (Vacant)`;

} else {

parkingSpaces[index].classList.remove("vacant");

parkingSpaces[index].classList.add("occupied");

parkingSpaces[index].textContent = `Space ${index + 1} (Occupied)`;

}

});

}

// Call the update function to initially display parking status

updateParkingStatus();

// Periodically update the parking status (you can fetch data from your system here)

setInterval(updateParkingStatus, 5000); // Update every 5 seconds

</script>

</body>

</html>

This code snippet creates a simple web page that displays three parking spaces and their status (vacant or occupied). It uses JavaScript to update the status every 5 seconds. In a real system, you would replace the simulated parkingData with data retrieved from your Raspberry Pi-based smart parking system. Additionally, you can implement user interactions, such as reservation and payment options, as needed.

Advantage

**1. Improved Efficiency:**

* **Reduced Search Time:** Users can quickly find available parking spaces, reducing the time spent searching for parking.
* **Optimized Space Utilization:** Parking operators can better manage and utilize available spaces, minimizing underutilized areas.

**2. Cost Savings:**

* **Reduced Fuel Consumption:** Users spend less time circling for parking, reducing fuel consumption and emissions.
* **Operational Efficiency:** Parking operators can optimize staff and maintenance costs.

**3. Convenience:**

* **Real-Time Information:** Users have access to real-time information on parking space availability, making parking more convenient and predictable.
* **Reservation and Payment:** Some systems allow users to reserve parking spaces in advance and make payments electronically, streamlining the process.

**4. Environmental Benefits:**

* **Reduced Traffic Congestion:** More efficient parking reduces traffic congestion and its associated environmental impact.
* **Emissions Reduction:** Less time spent idling in search of parking leads to reduced vehicle emissions.

**5. Data and Analytics:**

* **Data Collection:** The system collects valuable data on parking space utilization, helping operators make data-driven decisions.
* **Analytics:** Operators can analyze historical data to optimize pricing, resource allocation, and long-term planning.

**6. Scalability:**

* **Easy Expansion:** IoT-based systems are often scalable, allowing operators to add more sensors and parking spaces as needed.
* **Flexibility:** The system can adapt to changing parking needs, such as events or seasonal variations.

**7. User Experience:**

* **Navigation Assistance:** Users receive guidance to the nearest available parking space, enhancing their experience.

Disadvantages:

**1, High Initial Costs:** Implementing a smart parking system, including the installation of sensors, communication infrastructure, and software development, can be expensive. Smaller parking facilities might find it cost-prohibitive.

**2, Maintenance Costs:** Ongoing maintenance of the hardware components, such as sensors and communication devices, can add to the operational costs. Sensors may require periodic calibration or replacement.

**3, Technical Challenges:** Implementing and maintaining IoT devices, sensors, and communication networks can be technically challenging. Issues like connectivity problems, sensor malfunction, and software glitches can arise.

**4, Privacy Concerns:** Collecting data on parking space occupancy and user information raises privacy concerns. Users may be uncomfortable with their location and vehicle data being collected and stored.

**5, Reliability and Accuracy:** Sensor-based systems are highly dependent on the accuracy and reliability of the sensors. False positives or false negatives in parking space occupancy detection can lead to user frustration.

**6, Scalability:** Expanding the system to accommodate more parking spaces or additional features can be complex and costly. Ensuring a seamless user experience as the system scales can be challenging.

**7, Dependency on Technology:** A smart parking system is heavily reliant on technology, which means that any technical failures or outages can disrupt parking operations.

**8, Complexity for Operators:** Parking operators may find it challenging to manage and maintain the system, especially if they lack technical expertise.

**9, User Adoption:** Users may resist adopting the new system or find it difficult to use, especially if they are not tech-savvy.

**10, Limited Coverage:** Smart parking systems are more commonly found in urban areas or larger parking facilities. Rural or less densely populated areas may not have access to such technology.

**11, Power Requirements:** IoT sensors and communication devices require a power source. Ensuring a stable power supply can be a challenge, particularly in outdoor parking areas.

**12, Environmental Impact:** The manufacturing and disposal of electronic components and devices in the system can have environmental consequences.

**13, Data Security:** Protecting the collected data from cybersecurity threats is crucial. Hacking into the system can result in data breaches and misuse of user information.

**14, Integration Challenges:** Integrating the smart parking system with existing infrastructure or other smart city systems can be complex and may require custom solutions.

**15, User Experience Limitations:** The user experience can be compromised if the mobile app or web interface is not user-friendly or if the system does not provide real-time updates

**16, Barrier to Entry:** Small parking operators or municipalities with limited resources may face challenges in adopting and implementing smart parking systems.

Applications of smart parking systems:

1. **Urban Parking Management:**
   * Smart parking systems are commonly used in urban areas to efficiently manage and optimize on-street and off-street parking spaces. These systems reduce traffic congestion, enhance the utilization of available parking spaces, and improve the overall flow of traffic.
2. **Shopping Malls and Retail Centers:**
   * Shopping centers and retail areas use smart parking systems to provide shoppers with real-time information on available parking spaces, thereby improving the shopping experience. These systems can also offer reserved parking for VIP or loyal customers.
3. **Airports and Transportation Hubs:**
   * Airports, bus terminals, and train stations use smart parking systems to help travelers find parking quickly and reduce the time spent searching for a parking spot before their journey. These systems can integrate with travel booking services.
4. **Hospitals and Healthcare Facilities:**
   * Smart parking systems in healthcare facilities help patients and visitors locate parking spaces near the hospital entrance, ensuring timely access to medical services.
5. **Hotels and Hospitality:**
   * Hotels often employ smart parking systems to offer their guests a hassle-free parking experience. Some systems even allow guests to reserve parking spaces in advance.
6. **Smart Cities and Municipalities:**
   * Municipalities implement smart parking systems as part of their smart city initiatives to reduce traffic congestion, decrease emissions, and optimize the use of public parking facilities.
7. **Commercial Office Buildings:**
   * Office buildings with limited parking space can benefit from smart parking systems by maximizing the use of available parking spots and providing a better experience for employees and visitors.
8. **Universities and Educational Institutions:**
   * Educational institutions use smart parking systems to streamline parking for students, faculty, and staff. They may also employ dynamic pricing to manage parking demand during peak times.
9. **Stadiums and Event Venues:**
   * Event venues implement smart parking systems to accommodate large crowds efficiently, ensuring attendees can easily find parking during concerts, sports events, and other gatherings.
10. **Residential Communities:**
    * Gated communities, apartment complexes, and residential neighborhoods utilize smart parking systems to manage residents' and guests' parking needs, often integrating access control for added security.
11. **Tourist Attractions:**
    * Tourist destinations employ smart parking systems to enhance the visitor experience, making it easy for tourists to find and access parking spaces near popular attractions.
12. **Fleet Management:**
    * Businesses with vehicle fleets use smart parking systems to monitor and manage their parking facilities, ensuring that their fleet vehicles have convenient and secure parking options.
13. **Public Transportation Integration:**
    * Smart parking systems can be integrated with public transportation, providing passengers with information on available parking at transit stations and helping them make smooth transitions between driving and public transit.
14. **Environmental Impact Mitigation:**
    * In ecologically sensitive areas, smart parking systems can help manage the environmental impact by reducing traffic and parking congestion in sensitive zones.
15. **Car Sharing and Ride-Hailing Services:**
    * Car-sharing and ride-hailing companies can integrate with smart parking systems to reserve parking spaces for their vehicles and provide convenient pick-up and drop-off locations.

-----END-----