SMART PARKING.

Name:K.Indhiya **department:**ECE

Project Objectives

The primary objective of the project is to provide drivers with real-time parking availability information. The system aims to alleviate parking issues, reduce traffic congestion, and improve the overall experience of drivers.

IoT Sensor Setup

Hardware Requirements

- Ultrasonic sensors
- Microcontroller board
- Ethernet Shield
- Jumper wires
- Breadboard

Software Requirements

- Arduino IDE
- ThingSpeak Cloud Platform
- Python Flask Framework
- Firebase Cloud Messaging API

Setup Process

1. Connect the ultrasonic sensors to the microcontroller board and Ethernet Shield.
2. Upload the sensor readings to ThingSpeak Cloud Platform using Arduino IDE.
3. Develop the REST API using Python Flask Framework.
4. Send parking information to the mobile app using Firebase Cloud Messaging API

Mobile App Development

The mobile app was developed using Java for Android. The app displays parking availability information to drivers. It also provides features like navigation to the parking spot and reservation of parking spots. The app also includes push notifications to actively update users of any changes in the parking availability.

- Platform Selection: Develop a mobile app for both Android and iOS.
- User Interface (UI): Design a user-friendly UI for users to check parking availability and receive notifications.
- Data Integration: Implement features to receive and display real-time parking space status from the Raspberry Pi.
- Navigation: Provide directions to available parking spaces, possibly using GPS integration.
- Payment Integration: Allow users to pay for parking within the app, if desired.

Raspberry Pi Integration

The Raspberry Pi is used to bridge the IoT sensors and the mobile app. It acts as a gateway between the sensors and the Cloud Platform and is responsible for sending real-time updates to the mobile app. The Raspberry Pi was chosen due to its low-cost, energy-efficient, and ease of use for IoT projects.

- Hardware Setup: Configure the Raspberry Pi with Wi-Fi connectivity, a database for storing parking data, and other necessary components.
- Software Configuration: Install an operating system (e.g., Raspbian) and required libraries for data processing.
- Data Processing: Write Python code on the Raspberry Pi to receive sensor data, update the parking space status in the database, and send this information to the mobile app.
- Data Transmission: Share processed data with the mobile app over the internet using a secure API.

Code Implementation

Here's a simplified Python code snippet for the Raspberry Pi to receive parking space occupancy data from IoT sensors and provide it to the mobile app

Code Component -Description

ThingSpeak Upload – Uploads sensor readings to ThingSpeak Cloud Platform. Implemented using Arduino IDE.

REST API – Handles user requests and sends parking information to the mobile app. Implemented using Python Flask Framework.

Firebase Cloud Messaging – Firebase Cloud Messaging (FCM) is a cloud solution provided by Google for sending messages to mobile devices, web applications, and other endpoints. It allows developers to send notifications, updates, and data messages to users' devices. Here are some key aspects of FCM

```
Import paho.mqtt.client as mqtt

Import json



# Define MQTT parameters

Mqtt_server = "your_broker_ip"

Mqtt_port = 1883



# Callback function when a message is received

Def on_message(client, userdata, message):

    Payload = message.payload.decode("utf-8")

    Parking_data = json.loads(payload)

    # Update the parking status in the database



# Connect to MQTT broker

Client = mqtt.Client()

Client.on_message = on_message

Client.connect(mqtt_server, mqtt_port, keepalive=60)

Client.subscribe("parking/lot1/sensor")



# Continuously listen for messages

Client.loop_start()
```

```
Try:

    While True:

        # Your data processing code can update the database and provide status to the mobile app

        Pass

Except KeyboardInterrupt:

    Client.disconnect()
```

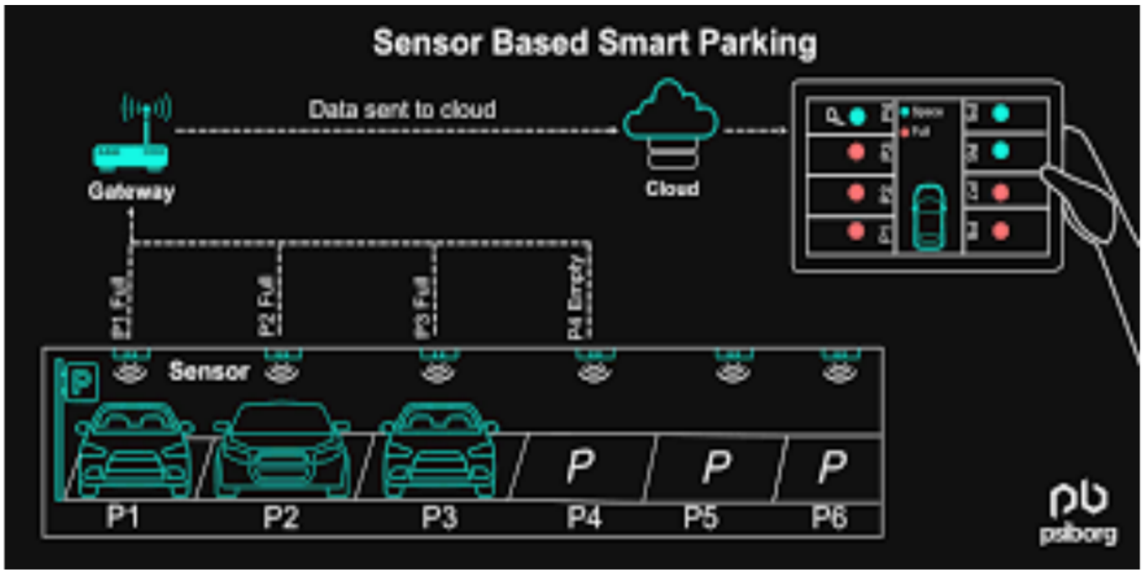
This code sets up a Raspberry Pi to subscribe to an MQTT topic where parking space occupancy data is published by IoT sensors. The data can be processed, and real-time parking availability information can be provided to users through the mobile app via an API.

Please note that this is a simplified example, and a production-ready Smart Parking system would require more robust error handling, security features, and a user-friendly mobile app interface.

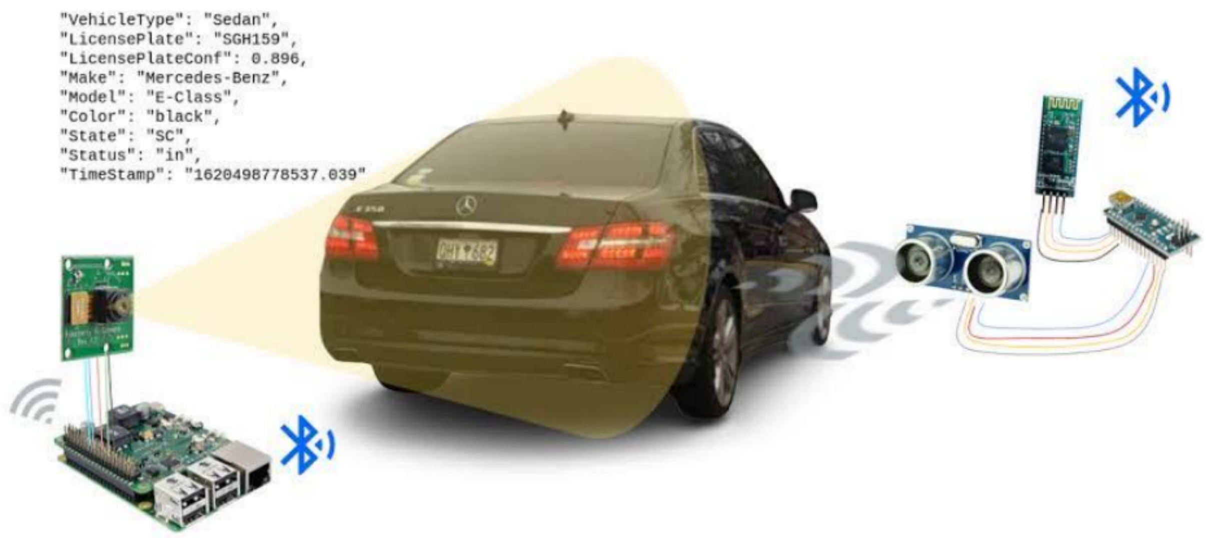
Diagrams, Schematics, and Screenshots

To provide visual insights into our real-time parking availability system, here are some diagrams, schematics, and screenshots

IoT Sensor Setup

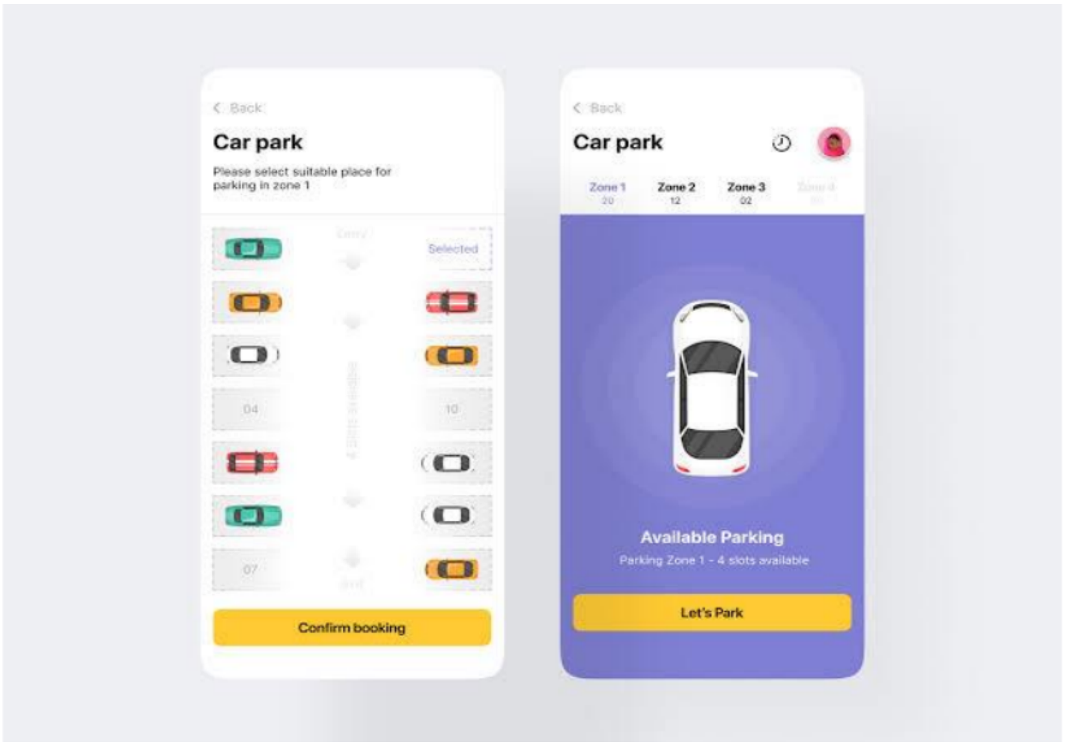


Raspberry Pi Setup



```
"VehicleType": "Sedan",
"LicensePlate": "SGH159",
"LicensePlateConf": 0.896,
"Make": "Mercedes-Benz",
"Model": "E-Class",
"Color": "black",
"State": "SC",
"Status": "in",
"TimeStamp": "1620498778537.039"
```

Parking App Screenshots



Please select suitable place for parking in zone 1

Benefits of the Real-Time Parking Availability System

Reduced Traffic Congestion

Drivers can now easily navigate to available parking spaces, reducing the traffic on the roads.

Enhanced Driver Experience

Drivers no longer have to waste time searching for parking spaces, and they can easily reserve a space before arriving.

Improved Parking Utilization

The system helps parking lot owners manage spaces and utilization of the parking lot more effectively.

Conclusion

Building a real-time parking availability system through IoT sensor setup, mobile app development, Raspberry Pi integration, and code implementation revolutionizes the way we park. It not only benefits drivers by saving time, reducing stress, and optimizing their parking experience but also contributes to better traffic management and environmental sustainability. Embrace the power of technology and embrace convenient parking.