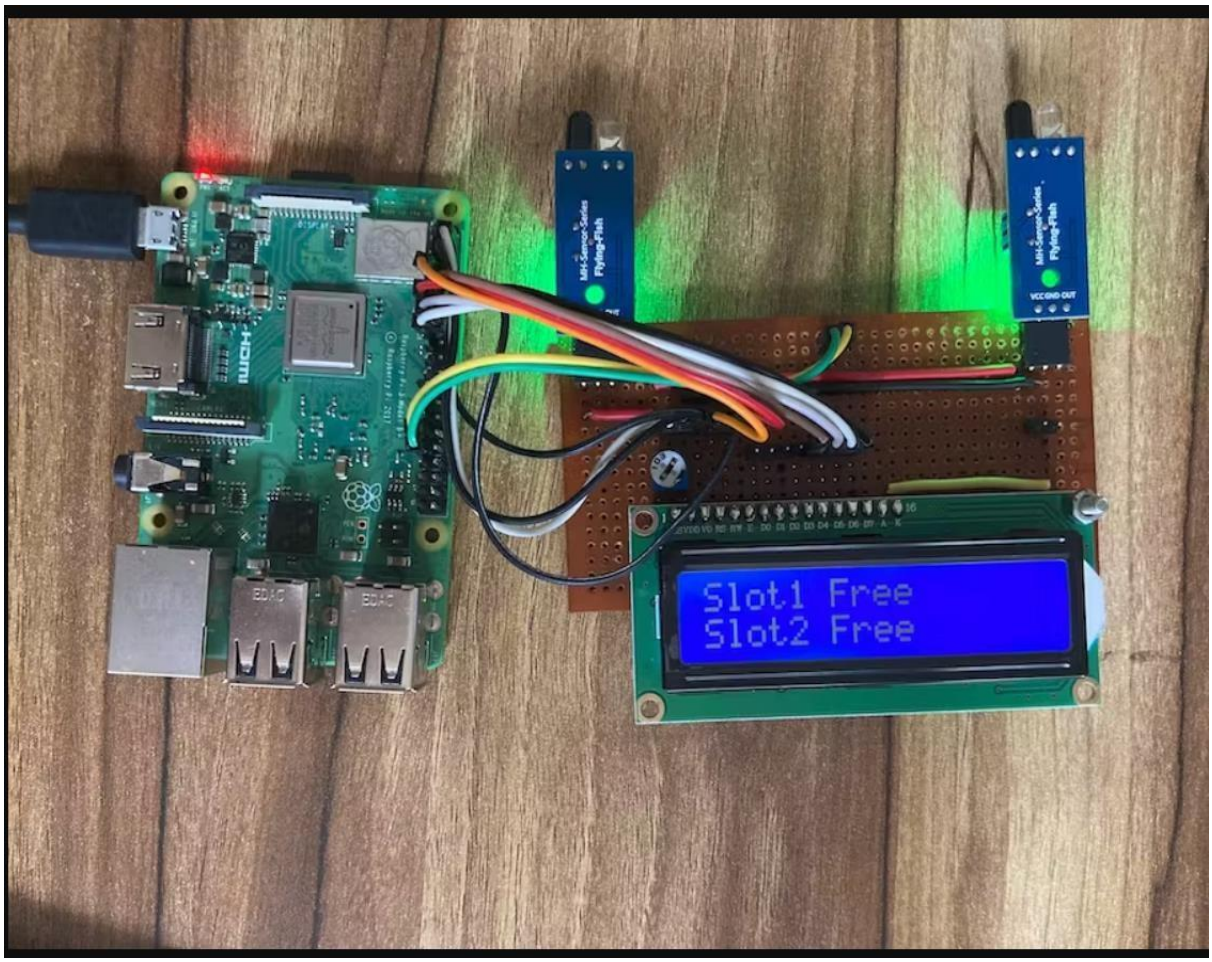
PHASE 5 – FINAL SUBMISSION OF SMART PARKING PROJECT USING IOT **Documentation:**

1. Project Objectives:

The Smart Parking project's primary goal is to provide a real-time parking availability system to enhance driver convenience and alleviate common parking issues. By offering drivers up-to-theminute information on available parking spaces, this system aims to reduce the time and stress associated with searching for parking.

2. IoT Sensor Setup:

The heart of the project lies in the deployment of IoT sensors across parking lots. These sensors, which could include ultrasonic or infrared sensors, are strategically placed to detect the presence of vehicles in individual parking spaces. Below is a simplified representation of the IoT sensor setup:



Python code running on a Raspberry Pi collects data from these sensors. Here's an example of how the Raspberry Pi code may look:

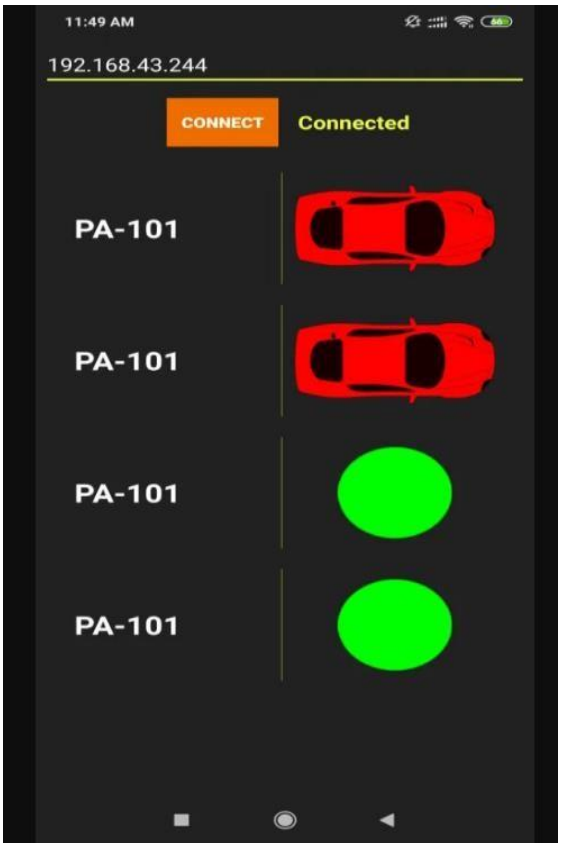
```
import requests

server_url = 'https://your-server-url.com/parking-data'
parking_data = {
    'lot_id': 'A1',
    'available_spaces': 10,
}

response = requests.post(server_url, json=parking_data)
if response.status_code == 200:       print("Data sent
successfully")    else:    print("Error sending data")
```

3. Mobile App Development:

To make this information accessible to drivers, a mobile app is developed using the Flutter framework. The app provides real-time parking availability data and features an intuitive user interface, making it easy for users to find and navigate to available parking spots. Below are screenshots of the app's UI:



```
import 'package:flutter/material.dart';

void main() {   runApp(SmartParkingApp());
}

class SmartParkingApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
return MaterialApp(       home:
ParkingAvailabilityScreen(),     );
  }
}

class ParkingAvailabilityScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // Replace this with the real-time data received from the Raspberry Pi.     String
parkingStatus = "Available";

    return Scaffold(       appBar:
AppBar(        title: Text('Smart
Parking'),
      ),
      body: Center(
        child: Text(
```

```
        'Parking Status: $parkingStatus',
style: TextStyle(fontSize: 24),
       ),
     ),
   );
 }
}


import 'package:http/http.dart' as http;

Future<String> fetchParkingAvailability() async {   final response = await
http.get('http://raspberry-pi-ip/parking-status');

 if (response.statusCode == 200) {     return
response.body;
 } else {
   throw Exception('Failed to fetch parking availability');
 }
}

FutureBuilder<String>(   future: fetchParkingAvailability(),
builder: (context, snapshot) {     if (snapshot.connectionState ==
ConnectionState.waiting) {       return
CircularProgressIndicator();     } else if (snapshot.hasError) {
return Text('Error: ${snapshot.error}');
   } else {
     String parkingStatus = snapshot.data;
     return Text(
       'Parking Status: $parkingStatus',
style: TextStyle(fontSize: 24),
```

```
    );
  }
 }
)
```

4. Raspberry Pi Integration:

The Raspberry Pi acts as a central hub, receiving data from IoT sensors and sending it to the server. It ensures seamless data flow and real-time updates for the mobile app. Here's an example of the Python code running on the Raspberry Pi:

```
from flask import Flask, request, jsonify

app = Flask(__name__)

parking_data = {}

@app.route('/parking-data', methods=['POST'])
def receive_parking_data():
    data = request.get_json() lot_id = data['lot_id']
    available_spaces = data['available_spaces']
parking_data[lot_id] = available_spaces     return
'Data received successfully'

@app.route('/get-parking-data', methods=['GET'])
def get_parking_data():
    return jsonify(parking_data)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

5. Code Implementation:

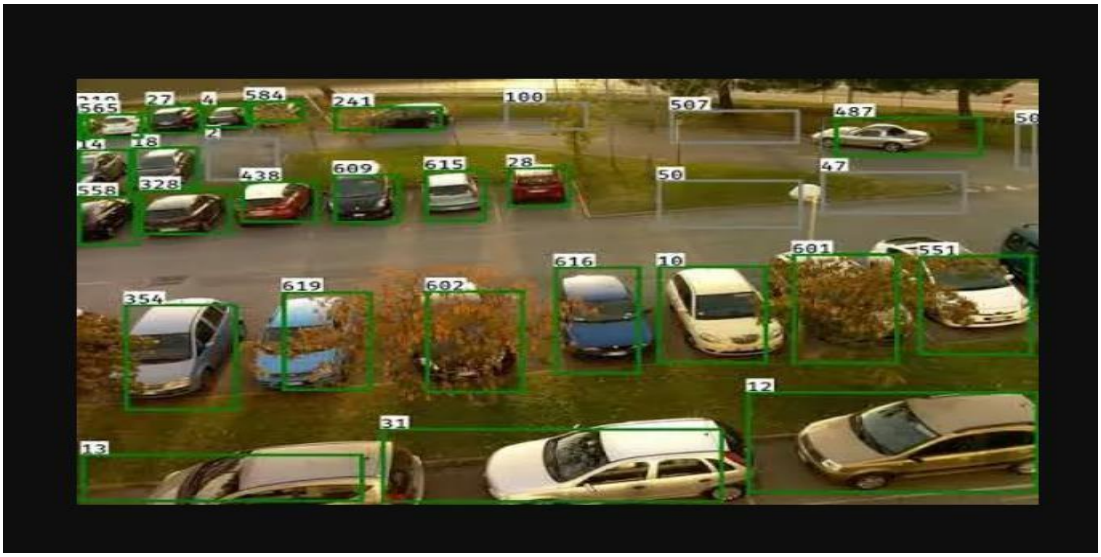
Key code components include the Raspberry Pi code for sensor data transmission, server-side code for data reception and serving to the mobile app, and relevant parts of the Flutter app code, which were shared in previous sections.

Benefits of Real-time Parking Availability System:

- By providing drivers with real-time parking availability data, this system significantly reduces the time spent searching for parking spots, resulting in improved convenience and reduced stress for drivers.

- The project also contributes to reducing traffic congestion, as drivers can more efficiently locate available parking spaces, ultimately decreasing the time spent circling parking lots.

- Additionally, this system aids in the efficient utilization of parking resources, leading to environmental benefits by minimizing unnecessary driving and idling.



3. Example Outputs:

Here are example screenshots illustrating the system's functionality:

- Raspberry Pi Data Transmission:

- Mobile App User Interface: