Project Title : Smart Parking

Phase 4 : Development Part 2

In this project, we will continue building a mobile app using the Flutter framework to display real-time parking availability data. However, instead of utilizing a Raspberry Pi, we will use an Arduino and the Wokwi platform for simulation. Additionally, we will incorporate ThingSpeak as the data storage and retrieval platform. This project's primary goal is to create a user-friendly mobile app that retrieves parking availability data from an Arduino connected to the Wokwi platform and presents it in real-time to users, all while leveraging ThingSpeak to store and manage the data. The app will provide valuable information to users, enhancing their parking experience by ensuring they have access to up-to-the-minute parking availability information.

Arduino data simulation and Thingspeak Communication:

```
#include <ThingSpeak.h>


// Variables for data and ThingSpeak configuration
const unsigned long Channel_ID = 1234567; // Your ThingSpeak Channel ID
const char* myWriteAPIKey = "ABCDE012345"; // Your ThingSpeak Write API Key
const int Field_Number_1 = 1;


int value_1 = 0;


void setup() {
  Serial.begin(115200); // For debugging in the simulation
}


void loop() {
  // Simulate data retrieval (you can use Serial input for data simulation)
  if (Serial.available() > 0) {
    value_1 = Serial.parseInt();
    Serial.print("Value received: ");
    Serial.println(value_1);
  }

  // Simulate sending data to ThingSpeak
  Serial.print("Sending data to ThingSpeak - Channel ID: ");
  Serial.print(Channel_ID);
  Serial.print(", Field: ");
  Serial.print(Field_Number_1);
  Serial.print(", Value: ");
  Serial.println(value_1);

  // Delay for simulation purposes
  delay(15000);
}
```

Code Overview:

The code in this section is responsible for simulating parking availability data and sending it to ThingSpeak, a cloud-based data platform. It uses serial input for data simulation, allowing you to emulate data retrieval. The code then sends this simulated data to a specific ThingSpeak channel. This Arduino component plays a crucial role in generating and transmitting parking availability data to be later fetched and displayed by the Flutter mobile app.

Thingspeak Integration:

- The code integrates the ThingSpeak library to facilitate communication with ThingSpeak for data storage and retrieval.

Parameter Configuration:

- Essential parameters such as Channel ID, Write API Key, and field number are configured for interaction with a specific ThingSpeak channel.

Data Initialization:

- Key variables, such as value 1, are initialized to manage simulated parking availability data. In this example, value 1 is set to 0.

Serial Communication Setup:

- The setup() function initializes serial communication at a baud rate of 115200, primarily for debugging in a simulation environment.

Data Simulation and sending:

- In the loop() function, data is simulated by checking the Serial input.
- Parsed data is printed to the Serial Monitor, and simulated data is sent to ThingSpeak by printing relevant details (e.g., Channel ID, Field Number).

Real-Time Parking availability in Flutter:

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

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

import 'dart:convert';


void main() {

  runApp(ThingSpeakApp());

}


class ThingSpeakApp extends StatefulWidget {

  @override

  _ThingSpeakAppState createState() => _ThingSpeakAppState();

}


class _ThingSpeakAppState extends State<ThingSpeakApp> {

  String value = 'Loading...';


  Future<void> fetchData() async {

    try {

      final response = await http.get(

        Uri.parse('https://api.thingspeak.com/channels/ Thingspeak_Channel_ID
/feeds.json?api_key=Your_API_key&results=1'),
      );

      if (response.statusCode == 200) {
        Map<String, dynamic> data = json.decode(response.body);
        setState(() {
          value = data['feeds'][0]['field1'].toString();
        });
      } else {
        setState(() {
          value = 'Error';
        });
      }
    } catch (e) {
      setState(() {
        value = 'Error';
      });
    }
  }

  @override
  void initState() {
    super.initState();
    fetchData();
  }

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: const Text('ThingSpeak App'),
        ),
        body: Center(
          child: Text('Value from ThingSpeak: $value'),
        ),
      ),
    );
  }
}
```

Code Overview:

This Flutter application is designed to provide real-time parking availability data to users by fetching information from ThingSpeak, a cloud-based data platform. The app offers a user-friendly interface for individuals searching for up-to-the-minute parking availability information, enhancing their parking experience. Through HTTP requests to ThingSpeak, the code ensures that users have access to the latest parking availability data, making it a valuable tool for anyone looking for convenient parking options.

Flutter App Initialization:

- The Flutter app is initialized with the main() function, starting the ThingSpeakApp().

Data fetching function:

- The fetchData() function makes an HTTP GET request to the ThingSpeak API to retrieve real-time parking availability data.

Data Processing:

- In the fetchData() function, the code checks the response status code and decodes JSON data if the response is OK.
- The app updates the value variable with the fetched parking availability data from the ThingSpeak feed.

Initialization and Widget building:

- The initState() function initializes data retrieval by calling fetchData().

- The built() function constructs the app's UI, including an AppBar and a Text widget to display real-time data.

Dynamic Data Display:

- The Flutter app dynamically updates the display with fetched data, providing users with up-to-the-minute parking availability information.

Conclusion:

In this project, we have successfully developed a mobile app using Flutter that provides real-time parking availability information to users. The app fetches data from ThingSpeak, a cloud-based data platform, and displays it in a user-friendly interface, enhancing the parking experience for individuals seeking parking solutions.