Iot phase 4 project

Problem statement : Use a mobile app development framework (e.g., Flutter) to create an app that displays realtime parking availability.

- Design app functions to receive and display parking availability data received from the Raspberry Pi.

Configure IoT sensors (e.g., ultrasonic sensors) to detect parking space occupancy. Write Python scripts on Raspberry Pi to collect data from sensors and send it to the cloud or mobile app server.

Submitted

By

N.Viswanath

S.Sanjai

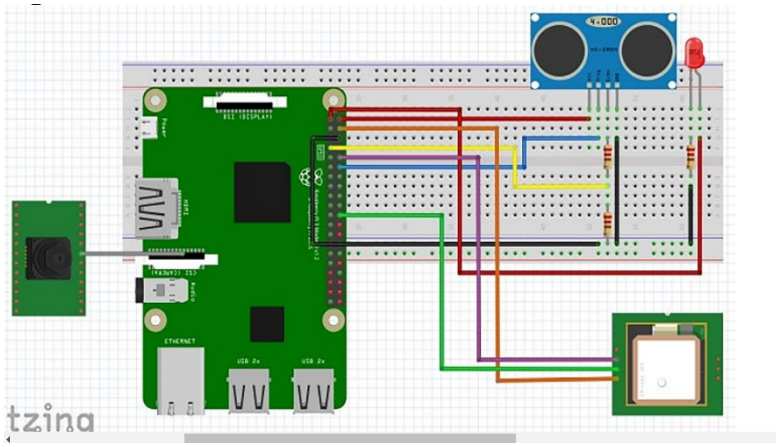
A.Ragunath

E.Periyadharshan

U.Karthikeyan

Pin connections:

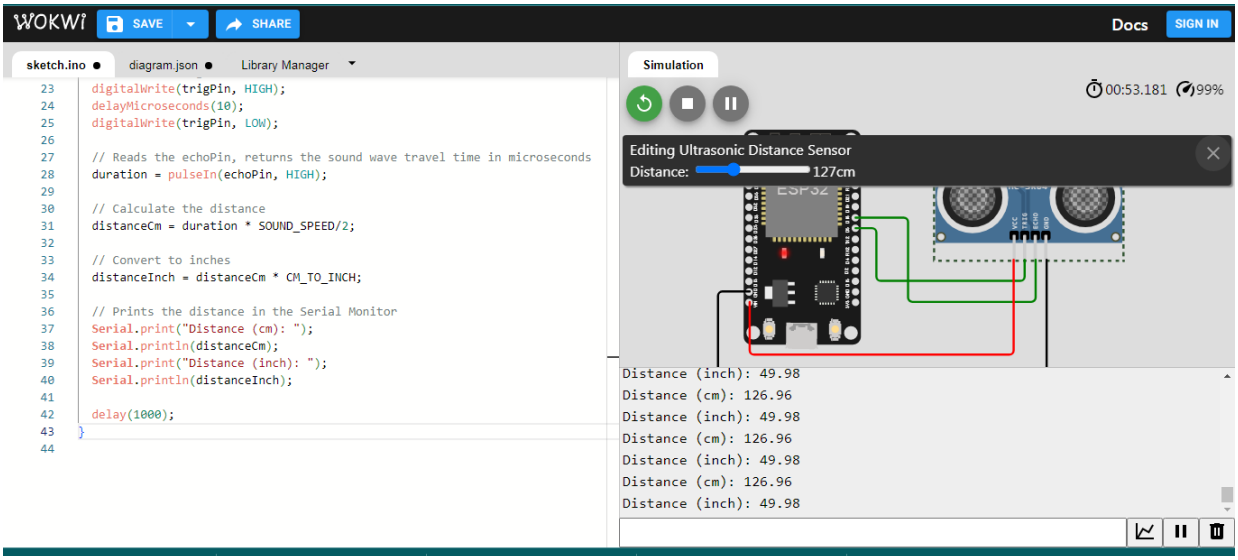
- **Ultrasonic Sensor**:
    - VCC (Power): Connect to 5V (Pin 2 or Pin 4) on the Raspberry Pi.
    - Trig (Trigger): Connect to a GPIO pin (e.g., GPIO17, Pin 11 on Raspberry Pi).
    - Echo: Connect to another GPIO pin (e.g., GPIO18, Pin 12 on Raspberry Pi).
    - GND (Ground): Connect to Ground (Pin 6, Pin 9, or Pin 14) on the Raspberry Pi.



Python script for parking detection using an ultrasonic sensor. This script measures the distance and detects if a car is in a parking space based on a distance threshold:

```
import RPi.GPIO as GPIO
import time

# Set GPIO mode and pins
GPIO.setmode(GPIO.BCM)
TRIG_PIN = 17
ECHO_PIN = 18
GPIO.setup(TRIG_PIN, GPIO.OUT)
GPIO.setup(ECHO_PIN, GPIO.IN)

def distance():
    GPIO.output(TRIG_PIN, True)
```

```
    time.sleep(0.00001)
    GPIO.output(TRIG_PIN, False)

    while GPIO.input(ECHO_PIN) == 0:
        pulse_start = time.time()

    while GPIO.input(ECHO_PIN) == 1:
        pulse_end = time.time()

    pulse_duration = pulse_end - pulse_start
    return pulse_duration * 17150

try:
    while True:
        dist = distance()
        print(f"Distance: {dist} cm")

        # Adjust this threshold as needed for your parking space
        threshold = 30  # Change this distance as needed
        if dist < threshold:
            print("Parking space occupied")
        else:
            print("Parking space available")

        time.sleep(1)

except KeyboardInterrupt:
    GPIO.cleanup()
```



Creating a real-time parking availability app involves multiple components: a Raspberry Pi to collect data, a backend server to process and send this data, and a mobile app to display it. In this example, I'll use Flutter for the mobile app development and assume you have a backend server set up to handle data transmission from the Raspberry Pi to the app. Here's a basic outline of how to create the app:

Prerequisites:

Flutter Installed: Make sure you have Flutter installed on your system. You can download it from the **official Flutter website**.
Backend API: Set up a backend API (using frameworks like Flask, Django, or Express.js) on your server to handle requests from the Raspberry Pi and serve data to the app.

Steps:

1. **Create a Flutter Project:**

```
flutter create parking_availability_app
cd parking_availability_app
```

2. **Add Dependencies:**

In your `pubspec.yaml` file, add the necessary dependencies:

yaml

Copy code

```yaml
dependencies:
  flutter:
    sdk: flutter
  http: ^0.13.3
  provider: ^6.2.3
```

Then, run `flutter pub get` to install the dependencies.

3. Create Models:

Create a Dart file to define your data models. For example, you might have a `ParkingAvailability.dart` file with a class representing parking availability data.

dart

Copy code

4. **Create API Service:**

Create a Dart file for your API service. This service will handle HTTP requests to your backend server.

dart

Copy code

5. **Create UI:**

Modify your `main.dart` **file to create the UI. Use the Provider package for state management.**

dart

Copy code

Replace `'YOUR_BACKEND_API_URL'` **with the actual endpoint where your backend server is running.**

6. Run Your App:

Connect your mobile device or emulator and run the app using the following command:

```
utter run
```

This basic app will request real-time parking availability data from your backend server and display it. Make sure your backend server is configured to handle these requests and send appropriate responses based on the data collected from the Raspberry Pi.