**Ringkasan Proyek: Vehicle Otonom (Autonomous Vehicle Navigation dengan Leaflet.js)**

### 🚀 Tujuan Proyek

Membuat kendaraan otonom sederhana yang dapat:

* Menentukan arah lintasan berdasarkan dua titik (start dan end)
* Menghitung derajat arah (bearing) antara dua titik
* Menghitung jarak antar titik di jalur rute
* Menentukan apakah belok kanan atau kiri lebih dekat
* Menampilkan visualisasi jalur dan info navigasi di peta interaktif menggunakan Leaflet.js

### 📂 Teknologi yang Digunakan

* **Leaflet.js**: untuk menampilkan dan mengelola peta interaktif
* **OpenStreetMap (OSM)**: sebagai sumber tile peta
* **OSRM API**: untuk mendapatkan rute dari titik A ke B
* **JavaScript (dengan jQuery)**: untuk logika client-side dan perhitungan arah

### ✅ Fitur yang Sudah Berhasil Diterapkan

1. **Klik 2 Titik di Peta**: Titik awal dan akhir.
2. **Ambil Jalur Rute Otomatis** dari OSRM.
3. **Hitung Arah (Bearing)** antar setiap titik sepanjang jalur.
4. **Hitung Jarak** antar setiap titik (dalam meter).
5. **Tentukan Arah Belok Lebih Dekat**:
   * Belok kiri jika delta negatif dan abs(delta) < 180
   * Belok kanan jika delta positif dan abs(delta) < 180
   * Sisanya sebaliknya
6. **Tampilkan Marker Biru** dan arah + jarak sebagai popup di tiap titik.

### 📃 Cuplikan Kode Penting

#### Fungsi Menghitung Bearing (0-360 Derajat)

function calculateBearing(lat1, lon1, lat2, lon2) {  
 const toRadians = deg => deg \* Math.PI / 180;  
 const toDegrees = rad => rad \* 180 / Math.PI;  
  
 const φ1 = toRadians(lat1);  
 const φ2 = toRadians(lat2);  
 const Δλ = toRadians(lon2 - lon1);  
  
 const y = Math.sin(Δλ) \* Math.cos(φ2);  
 const x = Math.cos(φ1) \* Math.sin(φ2) -  
 Math.sin(φ1) \* Math.cos(φ2) \* Math.cos(Δλ);  
 const θ = Math.atan2(y, x);  
 return (toDegrees(θ) + 360) % 360;  
}

#### Fungsi Menghitung Jarak (Meter)

function calculateDistance(lat1, lon1, lat2, lon2) {  
 const R = 6371000; // jari-jari bumi dalam meter  
 const toRad = deg => deg \* Math.PI / 180;  
  
 const dLat = toRad(lat2 - lat1);  
 const dLon = toRad(lon2 - lon1);  
 const φ1 = toRad(lat1);  
 const φ2 = toRad(lat2);  
  
 const a = Math.sin(dLat / 2) \*\* 2 + Math.cos(φ1) \* Math.cos(φ2) \* Math.sin(dLon / 2) \*\* 2;  
 const c = 2 \* Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));  
 return R \* c;  
}

#### Fungsi Menentukan Belok Kanan atau Kiri Lebih Dekat

function getTurnDirection(currentBearing, targetBearing) {  
 const delta = ((targetBearing - currentBearing + 540) % 360) - 180;  
 return delta > 0 ? "kanan" : "kiri";  
}

### 📌 Istilah Penting

* **Bearing**: Sudut arah dari satu titik ke titik lain relatif terhadap utara (0-360° searah jarum jam).
* **Polyline**: Garis penghubung antar titik dalam Leaflet.
* **OSRM**: Open Source Routing Machine, digunakan untuk mendapatkan jalur rute secara otomatis.
* **Tile Layer**: Lapisan dasar peta (dari OSM).

### 📄 Catatan Penting

* Radius marker biru kecil digunakan untuk tidak mengganggu visual rute utama.
* Pop-up info arah dan jarak digunakan untuk navigasi manual atau semi-otomatis.
* Titik awal dan akhir diatur via klik, cocok untuk demo sistem otonom.
* Cocok untuk diintegrasikan dengan **sensor fisik** (kompas digital, GPS, lidar).

### ✈️ Tahap Selanjutnya

* Integrasi dengan **sensor kompas real-time** dan **GPS module**
* Deteksi **rambu & obstacle** menggunakan **LiDAR** atau **ultrasonik**
* Mengatur navigasi otomatis berdasarkan heading aktual
* Mengintegrasikan feedback belok ke kanan/kiri berdasarkan hasil getTurnDirection()