Navigation 4-wheeler:

1. Introduction

This document describes the exploration, evaluation, and final selection of a navigation solution for a 4-wheeler instrument cluster based on the OKT507-C Linux board. The goal was to identify a reliable navigation platform that can be integrated into the vehicle display with good performance, offline/online support, and long-term scalability.

2. Objective

- Evaluate available navigation/map platforms suitable for automotive use
- Check feasibility on OKT507-C hardware and Linux environment
- Validate navigation rendering and interaction on the cluster display
- Finalize a recommended solution with clear installation and usage steps

2.1 Android Studio Installation Steps (Linux PC)

1. Check System Requirements

- 64-bit Linux
- Minimum **8 GB RAM** (16 GB recommended)
- At least **8 GB free disk space**
- Java is bundled (no need to install separately)

2. Download Android Studio

1. Open a browser
2. Go to **developer.android.com/studio**
3. Click **Download Android Studio**
4. Accept the license agreement
5. Download the **Linux (.tar.gz)** file

3. Extract the Downloaded File

Open **Terminal** and run:

```
cd ~/Downloads
tar -xvzf android-studio-*.tar.gz
```

This creates a folder named:

```
android-studio/
```

4. Move Android Studio to /opt (Recommended)

```
sudo mv android-studio /opt/
```

5. Start Android Studio

Run:

```
/opt/android-studio/bin/studio.sh
```

 Android Studio setup wizard will open

6. Complete Setup Wizard

1. Select **Standard** installation
2. Choose **Default theme**
3. Verify components:
    - Android SDK
    - Android SDK Platform
    - Android Emulator
4. Click **Finish**
5. Wait for SDK download

7. Verify SDK Installation

In Android Studio:

- Go to **File → Settings → Android SDK**
- Confirm:
    - SDK Platforms installed
    - SDK Tools (ADB, Emulator)

3. Platforms Explored

3.1 Mapbox

Overview
Mapbox is a widely used navigation and mapping platform providing SDKs for Android, Linux, and embedded systems.

Key Features

- Automotive-grade navigation SDK
- Online and offline map support
- Customizable UI and map styles
- Turn-by-turn navigation
- Supports GPU-accelerated rendering

Pros

- Good documentation and community support
- Flexible APIs for customization
- Suitable for embedded and automotive use

Cons

Requires account creation and access token Internet required for initial setup and map downloads

3.2 Mappls (MapmyIndia)

Overview
Mappls is an India-focused mapping and navigation platform with strong local data coverage.

Key Features

- Accurate Indian map data
- Navigation and routing APIs
- Android SDK support

Pros

- Excellent India-specific map accuracy
- Automotive partnerships

Cons

- Limited support/documentation for Linux and embedded boards
- SDK access requires approvals and licensing
- Integration on OKT507-C is not straightforward

4.Platform Comparison Summary

📌 Summary

Solution Type	Navigation	Offline	Embedded / Custom UI	License Free
Mapbox SDK	✔	✔*	✔	Partially
HERE Maps	✔	✔	✔	Paid/Free tier
TomTom	✔	✔	✔	Paid
Mappls	✔	✔	✔	Paid
OsmAnd	✔	✔	Limited	Open-source
Organic Maps	✔	✔	No	Open-source
CoMaps	✔	✔	No	Open-source
OSM + custom routing	✔	✔	Full	Open-source

*Offline depends on SDK/data setup.

5. Final Conclusion

After evaluating both platforms, Mapbox was selected as the preferred navigation solution for the OKT507-C based 4-wheeler cluster.

Reasons for Selection:

- Better compatibility with Linux and embedded systems
- Strong SDK and API documentation
- High flexibility for custom cluster UI integration
- Proven use in automotive and embedded products

6. Mapbox Setup – Installation & Login Steps

6.1 Account Creation

Use this web link: https://docs.mapbox.com/android/navigation/guides/install/

Steps to Create a Mapbox Access Token

Create / Log in to Mapbox Account

1. Go to mapbox.com

2. Click Sign up (or Log in if you already have an account)
3. Complete account verification (email, etc.)

Open the Access Tokens Page

1. After logging in, click your profile icon (top-right)
2. Select Account
3. In the left menu, click Access Tokens

Use Default Token (Quick Start)

- You will see a Default Public Token
- It usually starts with: **pk.eyJ1Ijo...**
- You can copy and use it immediately for development

Use Token in Application

Android

`xml file`

```
<string name="mapbox_access_token">pk.xxxxx</string>
```

6.2 SDK Installation (Android Navigation / Emulator)

Create an Android Project

1. Open **Android Studio**
2. Click **New Project**
3. Select **Empty Activity**
4. Choose:
    - Language: **Kotlin** (recommended)
    - Minimum SDK: **API 21+**
5. Click **Finish**

1.Add Mapbox Navigation Dependencies

Open `app/build.gradle` and add:

```
dependencies {
```

```
    implementation "com.mapbox.navigation:android:2.16.0"
}
```

Also ensure **Mapbox Maven repository** is added in `settings.gradle` or **project-level build.gradle**:

```
repositories {
    google()
    mavenCentral()
}
```

2.Add Mapbox Access Token

Add your Mapbox token in `AndroidManifest.xml` inside `<application>`:

```
<meta-data
    android:name="com.mapbox.access_token"
    android:value="pk.YOUR_MAPBOX_ACCESS_TOKEN" />
```

Use a **public token (pk.) only**

3.Add Required Permissions

In `AndroidManifest.xml`:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.INTERNET"/>
```

4.Initialize Navigation SDK

In your **Activity**:

```
val navigationOptions = NavigationOptions.Builder(this)
    .accessToken(getString(R.string.mapbox_access_token))
    .build()

MapboxNavigationProvider.create(navigationOptions)
```

5.Sync and Build Project

1. Click **Sync Now**
2. Resolve any Gradle issues
3. Run on **Emulator or Physical Device**

6.3 Common Issues & How to Fix Them (Android Navigation SDK)

1.Gradle Dependency Not Syncing

Problem:

- `Could not find com.mapbox.navigation:android`
- Sync fails

Fix:

- Ensure `mavenCentral()` is added in `settings.gradle`
- Check correct Navigation SDK version

```
repositories {
    google()
    mavenCentral()
}
```

2.Mapbox Access Token Error

Problem:

- App crashes on launch
- Error: `Mapbox access token is missing`

Fix:

- Add token correctly in `AndroidManifest.xml`
- Use **public token (pk.) only**

```
<meta-data
    android:name="com.mapbox.access_token"
    android:value="pk.YOUR_ACCESS_TOKEN"/>
```

3.lack Screen / Map Not Loading in OKT507-C

Problem:

- Navigation UI loads but map is blank

Fix:

- Enable Internet permission
- Check emulator network connectivity

```
<uses-permission android:name="android.permission.INTERNET"/>
```

4.Location Not Updating in OKT507-C

Problem:

- Blue dot not moving
- Navigation not starting

Fix:

- There OKT507-C board GPS is not there that why location is not updating.

5. App Crashes Due to Missing Permissions

Problem:

- Crash when starting navigation

Fix:

- Add runtime permission handling
- Ensure location permissions exist

```
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

7. Android Auto Evaluation

7.1 Desktop Head Unit (DHU) Setup in Android Studio

The Desktop Head Unit (DHU) is used to emulate an in-vehicle infotainment system for testing Android Auto applications.

Prerequisites

- Android Studio installed
- Android SDK installed
- USB debugging enabled on the test device or emulator
- Android Auto compatible app

Steps to Install and Set Up DHU

1. Open **Android Studio**
2. Go to **SDK Manager** → **SDK Tools** tab
3. Enable the following tools:
    - Android Auto Desktop Head Unit
    - Android SDK Platform-Tools
4. Click **Apply** and complete the installation

Locating the DHU Binary

After installation, DHU will be available inside the SDK folder:

<Android-SDK>/extras/google/auto/desktop-head-unit

Running the Desktop Head Unit

1. Open a terminal
2. Navigate to the DHU directory
3. Run the following command:

./desktop-head-unit

Connecting Device to DHU

1. Connect an Android phone to the system via USB (USB debugging enabled)
2. Run the following command to forward ports:

adb forward tcp:5277 tcp:5277

3. Launch Android Auto on the phone
4. The Android Auto UI will be displayed on the Desktop Head Unit window

Validation

- Navigation UI loads on DHU screen
- Map rendering and routing function correctly
- Touch and interaction events are reflected

7.1 Emulator Setup

- Android Auto was configured and tested using the Android Auto Emulator
- Navigation functionality was verified with Mapbox

7.2 Results

- Android Auto navigation working correctly on the emulator
- Maps, routing, and UI behaved as expected

8. Limitation with OKT507-C Board

Despite successful testing on the emulator, Android Auto is not compatible with the OKT507-C board due to:

- Android Auto dependency on Android Automotive / Android OS
- Android auto need google certification

- Lack of official Android Auto support for the hardware