

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

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

1. Visit the Mapbox official website
2. Create a Mapbox account
3. Generate an Access Token from the dashboard

6.2 SDK Installation (Android / Emulator)

1. Create an Android project in Android Studio
2. Add Mapbox dependency to `build.gradle`
3. Add the Mapbox access token to `AndroidManifest.xml`
4. Sync the project

6.3 Login & Token Configuration

1. Use the generated Mapbox access token
2. Configure the token in the application
3. Launch the application
4. Verify that maps load correctly

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 worked 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
- OKT507-C running Linux-based OS
- Lack of official Android Auto support for the hardware