

# **Navigation 4-wheeler:**

## **1. 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.Problem Statement – Navigation System for 4-Wheeler**

Most existing navigation integrations in embedded automotive clusters are limited to basic functionality such as:

- Rendering the map on the display
- Showing current vehicle position
- Displaying turn-by-turn instructions
- Fetching routes from an online service

## **3.Initial Approaches Attempted**

### **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

## 3. 3 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.

## **4.Finalized Approach:**

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

## **5. Mapbox Setup – Installation & Login Steps**

### **5.1 Account Creation**

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

### **5.2 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

### **5.3 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

## **6. Android Auto Evaluation**

### **6.1 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](path-to-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

## 6.2 Emulator Setup

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

## 6.3 Results

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

## 7. Challenges Faced

During the integration of Mapbox navigation on the OKT507-C Linux board and while maintaining the Ubuntu-based system, several technical challenges were encountered. These challenges affected stability, performance, and development continuity.

### 7.1 Network Dependency and Token Issues

- Mapbox requires:
  - Stable internet for initial setup
  - Valid access token for map loading

### 7.2 System Crash During Update Process

- The system crashed while performing:
  - **apt upgrade**
  - Kernel or graphics driver updates

Possible reasons:

- Power interruption during update
- Incompatible package versions
- Insufficient disk space during upgrade

This resulted in:

- Incomplete package installation
- Broken dependencies
- System failing to boot

## 8.Key Learnings:

### 8.1 Learning Android Studio and Android Application Development

Through this project, strong hands-on experience was gained in:

- Installing and configuring Android Studio
- Creating and managing Android projects
- Understanding the Android project structure
- Working with:
  - Activities and layouts
  - Gradle build system
  - AndroidManifest configuration

### 8.2 Understanding Mapbox Navigation SDK

The project provided practical exposure to the Mapbox platform, including:

- Creating a Mapbox account and generating access tokens
- Integrating Mapbox SDK into an Android project
- Displaying maps and handling map styles
- Implementing:
  - Current location tracking
  - Route drawing
  - Turn-by-turn navigation

## 9. Current Status of the POC

- Android Auto application launched correctly on the emulator
- Mapbox maps were rendered properly
- Route calculation and turn-by-turn navigation were functional
- User interaction through the Android Auto interface was stable
- When attempting to deploy the same POC on the **OKT507-C Linux board**, critical compatibility issues were observed.

## 10. Conclusion

Despite successful testing and validation of the navigation Proof of Concept on the Android Auto Emulator, the solution could not be deployed on the target hardware platform.

The primary reason for this limitation is the incompatibility between Android Auto and the OKT507-C board, which arises due to the following factors:

- Android Auto dependency on Android OS / Android Automotive OS  
Android Auto applications require a certified Android framework environment, which is not available on the OKT507-C platform.
- OKT507-C running a Linux-based operating system  
The target board operates on a standard Linux OS, and does not support the Android framework required by Android Auto.
- Lack of Android Auto certification for the hardware  
The OKT507-C board is not a Google-certified platform for Android Auto, which prevents official support and deployment.

As a result, although the navigation application worked correctly in the emulator environment, it is not deployable on the actual target hardware.

