In-Vehicle Location Based Services and Assessment on Embedded Systems

**Shahab Ahmed**

Geomatics (MSc) International Masters Student

*Karlsruhe University of Applied Science*

*Faculty of Information Management & Media*

*International Master of Science in Geomatics*



Table of Contents

[ABSTRACT 3](#_Toc528922314)

[1. INTRODUCTION 3](#_Toc528922315)

[2. OBJECTIVES 4](#_Toc528922316)

[3. THEORY & CONCEPTS 4](#_Toc528922317)

[3.1. Location Based Services 4](#_Toc528922318)

[3.1.1. Geolocation & Positioning 4](#_Toc528922319)

[3.1.2. Geovisualisation 5](#_Toc528922320)

[3.2. In Vehicle Location Based Services Infotainment 6](#_Toc528922321)

[3.3. In Vehicle LBS Features 7](#_Toc528922322)

[3.3.1. Navigation & Routing 7](#_Toc528922323)

[3.3.2. Connected Travel Services 8](#_Toc528922324)

[4. MATERIALS & METHODS AND SOLUTION APPORACH 9](#_Toc528922325)

[4.1. Study Area 9](#_Toc528922326)

[4.2. Work Flow & LBS Platform Consideration 9](#_Toc528922327)

[4.2.1. Data Collection & Preparation 10](#_Toc528922328)

[4.2.2. Location Software or Engine 10](#_Toc528922329)

[4.2.3. Deployment on Smart Core (Hardware) 17](#_Toc528922330)

[5. IMPLEMENTATION USING CASE STUDY 18](#_Toc528922331)

[6. REFERENCES 20](#_Toc528922332)

# ABSTRACT

*The purpose of this paper is to propose the master thesis project and provide a brief overview of how it can be successfully achieved. The goal of this study is integration of In-Vehicle location based services and develop an embedded navigation and connected travel services Infotainment to provide driver with relevant information to enable an efficient driving experience. The embedded navigation systems would have interactive and user friendly interface as well as it would also have advanced routing functionality i.e. dynamic routing based upon external real time traffic or road condition related data. The developed application is then deployed over Visteon’s Smart Core (A system on chip SOC) to analyze and evaluate its performance in terms of Geodata rendering i.e. assessing the performance of vector and raster data rendering and benchmarking based upon few other comparison parameters e.g. running the application on Smart Core in different operating systems like android and Linux, similarly benchmarking the performance of front-end mapping APIs like OpenLayers, Leaflet etc.*

# INTRODUCTION

The emergence of Information and Technology services (ICT) and the invention of Internet of Things (IoT) has altered the way devices interact with each other. As a result, Location-Based Services (LBS) have also recently emerged as an active area of research *[1]*. The studies have shown that LBS are playing an important role in automotive industry also. This field of research and development is gaining more and more attention because it is influencing the automotive businesses in a positive way.

In-Vehicle Location Based Services (LBS) e.g. embedded navigation and connected travel services are the features which drivers value the most while buying a new car. These location aware and driver centric features provide drivers with relevant travel information enabling an efficient driving experience *[2]*. An optimized and informative navigation to driver’s destination is one of these features. In addition to that, travel services i.e. finding parking spaces as driver approaches the destination, nearby available fuel stations in case of low fuel warnings, live weather reports for the end destination, on any point or along the route enabling faster and safe routing etc. are all the key travel features in now a days modern in-vehicle navigation system infotainments.

This study is focused on both research and application of Location based technologies and knowledge to the in-vehicle infotainment development for efficient driving assistance using embedded navigation for optimal dynamic routing and other connected travel services. The optimization of the routing can be achieved by minimizing the cost of routing as much as possible. For that, the concept of dynamic re-routing will be incorporated i.e. intelligent routing based upon dynamically avoiding traffic congestion, construction measurements or any other obstacle causes etc. This data will be supplied dynamically to the routing system to enhance and optimize the routing.

Furthermore, in this study, the navigation system infotainment developed will then be deployed and tested on Smart Core which is a system on chip developed by Visteon, and it is responsible for operating different guests embedded in the vehicle on a single chip by means of virtualization. These guests are infotainments, Clusters and Telematics etc. The developed in-vehicle navigation and connected services infotainment will be one of the guests to be operated on the embedded system which is Smart Core. So, it is important to analyze the performance of the system. The testing and evaluation of the navigation system’s performance on Smart Core will be dependent on various criteria’s. These use case criteria’s can be based on comparison between operating systems, formats of Geodata being rendered, front end mapping API’s etc.

# OBJECTIVES

The aims and objectives of this study are to explore usage of location based services in automotive domain:

* study and implement static and dynamic routing problems and potentials
* explore modern navigation systems
* Develop an in-vehicle location based services infotainment that has navigation systems and other connected travel services for efficient user experience.

Moreover one of the main objectives of this study is to develop both Online and Offline map-based navigation applications for android. Both applications will then be installed and run as infotainment on embedded system called Smart Core. The performance of both applications mainly in terms of geo data rendering on smart core will be analyzed and evaluated using profiling tools. The performance evaluation will be based upon certain resources or components of embedded system e.g. GPU, CPU, Random Access Memory (RAM), Network speed (in case of online application) etc. This would give a general overview of the system and would also help to better analyze and benchmark the Smart Core and to predict system requirements for running any other application in the future.

# THEORY & CONCEPTS

## Location Based Services

Location Based Services (LBS) is a modern concept that can be defined as *“services that integrate a mobile devices’ location or position with other information so as to provide added value to a user [3]”.* These services are accessible with mobile devices over mobile network.

With the advent of mobile communication technologies, rapid development of information and there integration with mobile devices has now become a common exercise. The technologies like Geographic Information Systems (GIS), Global Positioning Systems (GPS), radio frequency identification, and many other location identification technologies with varying costs and accuracies are now making the use and practicing of LBS applications very common. The common examples of LBS applications include:

* Travel services e.g. Routing, city, hotel and restaurant guides
* Emergency e.g. Locating a person in need, warn persons in danger
* Fleet management
* Phone tracker

### Geolocation & Positioning

Geolocation refers to the identification of real world geographic coordinates or location of an object. It is one of the core components of a LBS service or an application. The mobile devices or vehicles are equipped with positioning systems which are used to locate the vehicle on the earth to be able to retrieve the navigation and related travel services based on that location.

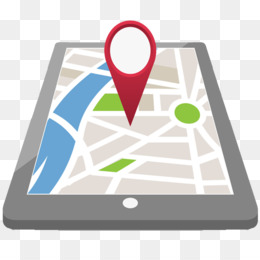


Figure 3.1: Geolocation [4]

There are various positioning systems and technologies being used in today’s modern In-vehicle location based services. Generally the positioning systems can be categorized in two type’s i.e. global and local positioning systems:

**Global Systems** consist of Global Navigation Satellite System (GNSS). GNSS refers to constellation of satellites in space that send signals from space that transmit positioning and timing data to GNSS receivers and these receivers then use this data to calculate and determine the location. USA’s Navigation Satellite Timing and Ranging (NAVSTAR) Global Positioning System (GPS) is one of the renowned examples of GNSS, which is also used as assisted GPS or shortly A-GPS. Other examples of GNSS include Europe’s Galileo, Russia’s Global'naya Navigatsionnaya Sputnikovaya Sistema (GLONASS) and China’s BeiDou Navigation Satellite System [18]. In addition to GNSS, other global systems of positioning involve Cell-ID technology using mobile networks etc. while on the other hand **local Systems** involve low range positioning technologies e.g. Bluetooth, RFID, WiFi etc.

GPS was developed in 1973 and is a most commonly used positioning system. It enables 3 dimensional positioning near the earth by measuring the runtime of signals between the satellite and the GPS-receiver, from which the distance and the position can be deduced [5].

Over the centuries, various kinds of technologies have been tried for navigation. The discovery of GPS has changed the face of modern navigation forever. In modern vehicle navigation, the second-generation guidance system developed by the U.S. Department of Defense in the mid-1980s, known as the NAVSTAR GPS, is becoming widely used [3].

The A-GPS technology is being extensively used with GPS-capable cellular devices to optimize and fasten the positioning. In stand-alone positioning systems e.g. Dead reckoning that use relative positioning, the GPS can be used as secondary or assisted system for providing absolute positioning, hence enhancing and improving the positioning.

As the internet becomes rapidly mobile because of the increasing mobile devices connected to internet, location based internet services and application have taken much more prominent role. These applications or services rely on these global and local positioning systems. Day by day there are more developments and improvements in geolocation and positioning. With the release of GNSS raw measurements on Android’ Nougat and higher versions, the European GNSS agency (GSA) has launched a Task Force for GNSS Raw Measurements to participate with leading professionals and authorities in navigation and positioning, and boost innovation around this newly released feature of GNSS [19].

The accuracy of these positioning systems is also very important because LBS uses accurate and real time positioning systems and GIS to locate the moving object and the information provided by these systems is sensitive for the current real position of the user and can be used to guide and inform users about current geographical conditions e.g. real time traffic and weather conditions related to current position of user.

### Geovisualisation

Geovisualisation is a vital component of modern In-vehicle Location based services and navigation systems and is also a critical part to be focused in this research because ‘what you see is what you get’. Geovisualisation is an art of visualization and according to Kraak & MacEachren, the definition of geo-visualization is *“Geo-visualization can be described as a loosely bounded domain that addresses the visual exploration, analysis, synthesis and presentation of geospatial data by integrating approaches from cartography with those from other information representation and analysis disciplines, including scientific visualization, image analysis, information visualization, exploratory data analysis and GI Science”* [6]. In LBS there are various devices i.e. smartphone, PDA, Tab, GPS etc. that use Geovisualisation to visualize real time information about the user’s location and its surroundings. In this study, since the LBS platform is vehicle or car, so the Geovisualisation will be realized through In-Vehicle infotainment and will be discussed later.

## 3.2. In Vehicle Location Based Services Infotainment

Location based services as infotainment services in vehicles and cars is another most important part of study because ultimately this would be front end or the user end which is responsible to meet the user requests in availing the desired location related services e.g. navigation and other travel services etc.

*“An In-vehicle infotainment (IVI) or in-car entertainment (ICE) is an infotainment offer that can be used in motor vehicles by the driver and passengers. Infotainment offers are for entertainment, information, and communication and driver assistance.”* [7]

An IVI is an information and broadcasting service inside the vehicle to broadcast digital multimedia as well as to inform the driver about conditions of traffic, weather etc. They are concerned with navigation using some positioning systems e.g. GPS [7].

These infotainment systems inside the vehicle are the most suitable platform to deliver location or space related services inside the vehicle. These kind of location based services are also called infotainment services which use pull-based model of LBS i.e. location information is either transmitted with request or retrieved as service [3].

The vehicles can use some systems or devices to deliver location based or infotainment services to the driver. The pictorial representation of these systems can be seen in the Figure 3.2. The In-Vehicle LBS are delivered by telematics systems, portable navigation device (PND) and as navigation or mapping apps on mobile devices.

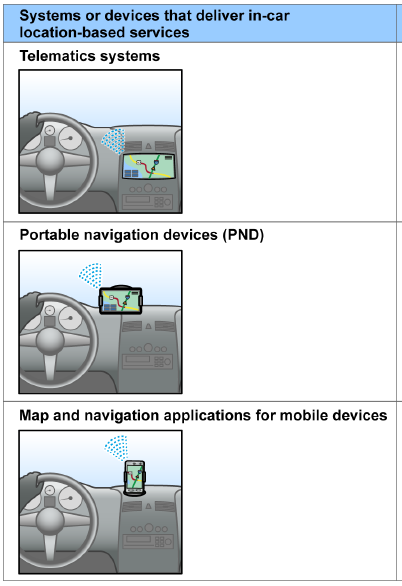


Figure 3.2: In-Vehicle LBS Systems/devices [8]

Telematics systems are often provided by Car manufacturers. The consumers or drivers retrieve these services using devices embedded in their vehicle or via their mobile devices or phones which are connected to their vehicles. The examples of these systems can be General Motors’ OnStar, Ford Sync, and Chrysler UConnect etc. The PNDs are provided by specific PND manufacturers or companies e.g. TomTom, Garmin etc. PNDs are equipped to transmit location data directly to consumers as services or through mobile devices of consumers that are connected to PNDs. The third system is mobile devices having Map and Navigation applications provided by mobile app developers. These services are received through smart phones [8]. The examples of such apps are Google Maps, Scout GPS Navigation.

## 3.3. In Vehicle LBS Features

One of the most common examples of LBS applications include emergency services, tour planning, car navigation systems etc. This section will emphasize on theory and concepts of those features and functionality which are supposed to be the part of the desired In-vehicle location based services infotainment e.g. Navigation systems and other connected travel services, they are described below:

### Navigation & Routing

Modern Navigation system is an integrated collection of position, orientation sensors, computing and communication hardware and software which are used to guide the movement of vehicles and other objects. The ease of availability of positioning technologies like GPS and other location determining technologies e.g. mobile networks etc. have changed the face of the modern navigation [3]. Navigation system provide the driver the opportunity to be guided to his destination, by means of spoken or visual advices. To accomplish this, the driver first has to enter his destination into the system. The example of the destination could be a city center, an entire street, or an address including a house number. Information on every possible destination is contained in a database that is stored on a CD, DVD, SD-card or internal memory of the embedded system that has to be inserted into the car navigation system. Also data from external providers about interesting places such as hotels, restaurants or other points of interests is stored on the local storage of the system or can be fetched from any network/internet in case of modern connected navigation systems. The driver can ask for a list of all hotels in a certain area for example [31].

Since the navigation is process of guiding the movement from one place to another, therefore it has to be accurate and intelligent enough. In this study more research and focus will be applied to location engine i.e. LBS software and its intelligence as well as Computing platform (Smart Core) rather than the positioning systems & technologies.

The key components of a car navigation system are positioning, i.e. determining the current position of the car on the road network, route planning, i.e. planning a route from the position of the car to the destination, and guidance, and i.e. giving instructions to the driver [31].

Route determination is a component of navigation system along which guidance or navigation is provided. Routing is searching the best way in a weighted graph, where weighting is done in terms of cost which could be anything e.g. distance, effective velocity, energy consumption, travelling time etc. [5].

Routing can also be defined as “*the technique of calculating the optimal course, based on specific criteria, between an origin and destination*”. *Route information consists of a graphical representation of the route and a detailed turn-by-turn route description. It enables mapping applications to render the geographical representation of the route together with the map data, so that the route is displayed on the map [10].* The Figure 3.3 shows an example of optimal route calculation between start and destination points.

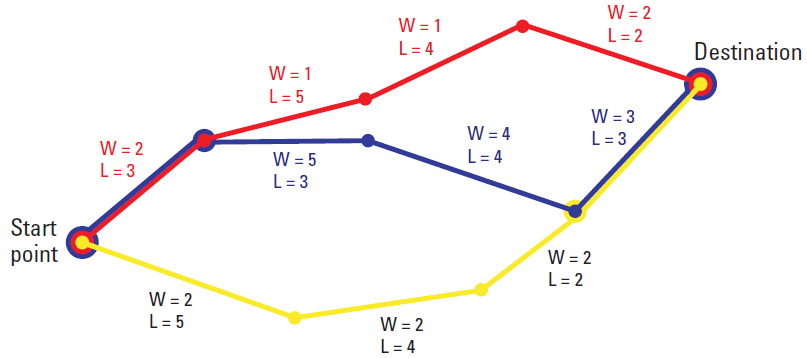


Figure 3.3: Route calculation with link-

-lengths (L) and road weightings (W) [9]

This optimal course is the best route which can be shortest, fastest, fewest turns or non-freeway route, depending upon the criteria being applied [3].

#### Routing Algorithms

The routing engine or software in navigation systems use a technique called routing algorithm to calculate the optimal route. *“Algorithms for route planning in transportation networks have recently undergone a rapid development, leading to methods that are up to three million times faster than Dijkstra’s algorithm. We give an overview of the techniques enabling this development and point out frontiers of ongoing research on more challenging variants of the problem that include dynamically changing networks, time-dependent routing, and flexible objective functions” [11].* The routing algorithms work on graph theory to search for paths by traversing from one node to another along the edges of a graph. There are a lot of other algorithms available but in this research, only two algorithms will be focused or used i.e. Dijkstra and A\*.

#### Static Routing

In static routing same route is given irrespective of the current traffic conditions. Vast majority of navigation systems use static routing [9]. Provided the road network as graphs having n nodes and m edges, then a shortest-path query between a source node s and a target node t will ask for a minimum weight d(s, t) of any path from s to t. In case of static routing, the edge weights do not change [11]. In static routing, drivers do not have knowledge of current or real time traffic conditions and this might guide drivers to some congested or to any no-go areas.

#### Dynamic Routing

As in case of static routing there is no knowledge of whether the calculated route would be better or not, this gave rise to demand of dynamic navigation systems having dynamic routing functionality which is becoming popular. Dynamic systems use real time data to support derivation of new route or to inform the about the problems ahead [9]. The dynamic data supplied to navigation systems can be related to traffic congestions, construction, road accident or any other kind of obstacles which can cause delay and can be avoided using dynamic routing. Figure 3.4 shows an example of congestion and delay in the calculated route and the dynamic navigation system prompts to the user for an alternative optimal route.

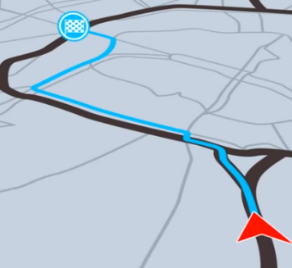
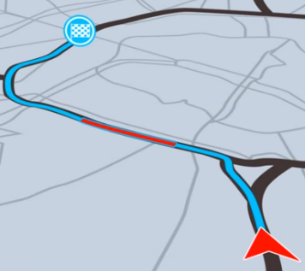


Figure 3.4: Example of road congestion in calculated route and an alternative dynamic route [12]

These dynamic routing systems are being used in modern navigation systems and are also being used by many auto and navigation systems manufacturers. In this study dynamic routing will also be a core component of navigation system to be developed as part of in-vehicle location based services.

### Connected Travel Services

In addition to navigation and routing, connected travel services are another important feature of In-vehicle LBS. These travel services provide relevant and useful information to drivers and let them have a more informative and a comfortable journey. These services include updates about weather, traffic situations, nearby fuel stations, finding restaurants or hotels etc. These services are valued by the people while purchasing vehicles with these embedded services. In this study this feature of connected travel services will be also developed and integrated.

Dynamic data such as traffic and live weather Reports can influence the journey and can support and inform the driver along the route. Moreover proximity searches of points of interest e.g. restaurants, parking spaces, fuel stations are important aspect to be added in this study as travel services based on location.

# MATERIALS & METHODS AND SOLUTION APPORACH

## Study Area

The geographical study area for the road networks and other relevant geospatial to be used in this study will be limited to only Baden Württemberg state of Germany. Baden Württemberg is located in Southwest of Germany. With a surface area of 35,751 square kilometers and population of around 10.8 million people, Baden Württemberg is considered as the third biggest state of Germany [13].

There is also a study conducted on mobility in Baden Württemberg and concentrates on different ways of transformation towards a sustainable mobility. The study shows that today's transport system, its infrastructure and our mobility behaviors are resource intensive and cause significant strain on humans, environment and climate. The primary goal of the project was to develop scenarios for sustainable mobility in 2050, then review these scenarios for their true sustainability. The Baden-Württemberg Foundation and the state Baden-Württemberg wanted in the study to show different ways of how mobility can develop in the future and the objectives of an ecological, economic and social sustainability of the mobility system can be achieved. Achieving the goals requires a fundamental change [20].

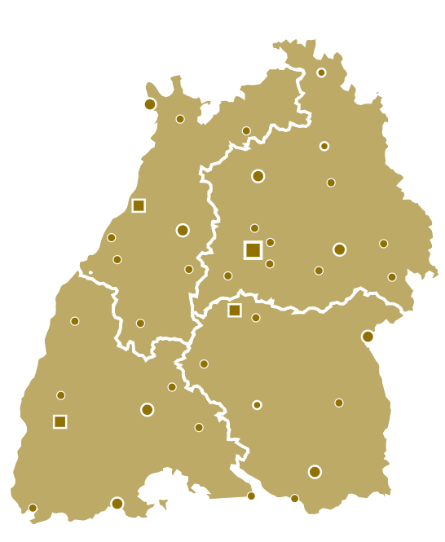


Fig. 4.1: Baden Württemberg [21]

## Work Flow & LBS Platform Consideration

This section focuses on consideration of platform for the LBS application and also designing the flow of whole work in order to proceed. In any LBS platform, it is critical to evaluate and assess the two areas that are related to the quality of LBS application, which are Data and LBS software or engine. In addition to these two, there is another important area to be considered in this study, which is the computing platform called ‘Smart Core’.

### Data Collection & Preparation

The source of geospatial data will be OpenStreetMap (OSM) available on Geofabrik website [14]. *“The OSM is an international project founded in 2004 with the goal of creating a free world map. For this we collect data worldwide about roads, railways, rivers, forests, houses and much more”* [15]. The street or road networks will be the primary dataset from OSM to be used in this study for routing and navigation purposes. OSM will also be the source of various points of interests (POIs) to be used in the LBS software e.g. destination points in navigation, nearby fuel stations, restaurants etc. Similarly data for dynamic routing e.g. traffic congestion, obstacles etc. will be supplied externally either through any online API like MapQuest Traffic API or it will be just a sample test data.

The LBS application will typically use information and data from three different content databases that are:

ROAD NETWORK DATABASE

This database contains all types of street networks as well as other related digital geographical data of Baden Württemberg region. A spatial database management system will be selected to manage and store the geospatial data. The geospatial data will be stored in vector format which is composed of links or edges representing the roads and connecting points represent intersection or other road features. These links and connecting points define the geometry of the data. In addition to geometry, the data also has associated attributes of these road features e.g. one way streets, exit signs, forbidden turns, bridges, tunnels, car-height restrictions etc. Furthermore the road network formed by these edges and nodes will be prepared for network analysis i.e. routing.

POINTS OF INTERESTS

Point of Interest (POI) information is another important content database of this LBS application. The POI information helps drivers locate businesses near a specified location ‘Where is the restaurant?’ or ‘where is the nearest fuel station?’ The information related to answering of these questions is compiled into POI map databases.

POI information is used in one of the most popular LBS applications called as Yellow Pages, or concierge services [3]. The integration of POI database with road databases, creates a very detailed digital representation of the street networks and business services along them.

DYNAMIC DATA

Dynamic data is the third data component of this LBS application which refers to the supporting data to override existing map data. In optimal routing dynamic data plays an important role by supplementing existing static map information e.g. traffic accidents, jams, road construction, weather condition can all affect road’s capacity or its supported traffic speed. Our LBS software will be well designed to work with dynamic data.

### Location Software or Engine

In location aware systems, the Location software or engine is considered as the heart of the system. It consists of the software components that add intelligence to processing and usage of geospatial or map data. Like the quality of data being used, the quality of these software components is also important to generate accurate results in the LBS system. Following are the key functionalities required in our in-vehicle location engine:

NAVIGATION AND ROUTING

The navigation and routing functionality is a core requirement in our LBS system which would be responsible for enabling the user/driver to obtain a route between two points and get driving directions and guidance to reach to the destination point.

PROXIMITY OR NEIGHBORHOOD SEARCHES

The functionality of proximity searches will use POI database information to find the fuel stations, restaurants and other businesses or landmarks in neighborhood of a specified location.

GEOCODING & REVERSE GEOCODING

Like all other geospatial applications, geocoding and reverse geocoding will also be the features of this LBS application. Geocoding is conversion of a geographical address to its geographical coordinates i.e. latitude and longitude and reverse geocoding is opposite of that. A geocoding API which is compatible with OSM, will be used to add the geocoding functionality in this application.

USER INTERFACE

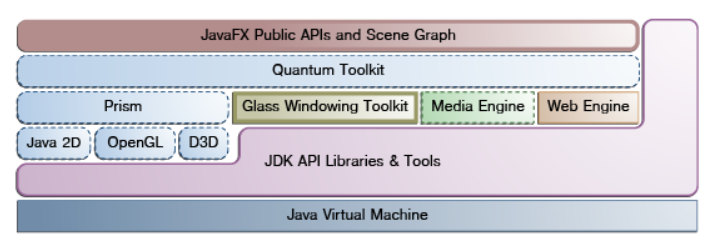
The User Interface (UI) for this LBS application is in-vehicle infotainment running as one of the guest platform of Smart Core. The infotainment is one of the cockpit-electronics products, which is running on single, multi-core chip, and is accessible through integrated Human Machine Interface (HMI). The UI will be interactive to enable the user (driver) has a great and efficient driving experience. The key features of the UI will be:

* Find the desired locations by name or address
* Navigate to these locations using a turn by turn navigation instructions in 3D/tilted mode.
  + - 1. **Development Environment**:

This section discusses the environment and software development kit (SDK) used to develop our location engine or software. The SDK involves web based technologies e.g. JavaScript libraries for building cross platform mapping interface. The other tools in SDK are standalone desktop based software used in the development of our Location software e.g. Geoserver, Spatial database management system called PostGIS, Smart Core as embedded computing platform and various other utilities discussed below.

**JavaFx**

JavaFx will be used for building native UI and integrating web-based map view (built using OpenLayers-A JavaScript library) into it using its web component. *“What Is JavaFX? JavaFX is a set of graphics and media packages that enables developers to design, create, test, debug, and deploy rich client applications that operate consistently across diverse platforms”* [22]. JavaFx is a set of Java libraries and the code of JavaFx based applications can reference APIs from any Java library e.g. For example, “*JavaFX applications can use Java API libraries to access native system capabilities and connect to server-based middleware applications*” [22]. Following figure shows the architectural components of the JavaFx and also shows that how they are interconnected:



2Fig. 4.2: JavaFX Architecture Diagram [23]

As shown in the Fig. 4.2. Below the JavaFX public APIs, there is an engine that is responsible for executing your JavaFX code. This engine is further composed of subcomponents that include the new JavaFX high performance graphics engine, called Prism; the new small and efficient windowing system, called Glass; a media engine, and a web engine [23].

The web component is one of the most important components of JavaFx as well as reason for using JavaFx as front-end UI building library in our Location software. The web view provided by the web component uses WebKitHTML technology to make it possible to embed web pages within a JavaFX application. Java APIs can call JavaScript running in WebView and JavaScript running in WebView can call Java APIs [22]. Using this key feature of JavaFx, our Location software will be able to make best of both Java and JavaScript environments and will provide high quality software with interactive UI.

**OpenLayers**

OpenLayers is a JavaScript Mapping API for front end map visualization. It is feature-packed and high-performance open source library for creating interactive web maps. It can display map tiles, markers or POI, vector data etc. The vector layers of OpenLayers can render vector data in various formats e.g. GeoJSON, TopoJSON, KML, GML, Mapbox vector tiles and other formats [24]. In our Location Software, OpenLayers will be used to render road networks as basemap (in both raster and vector formats) and it will also render the routes between two different points in GeoJSON format.

**Ol-Cesium**

Ol-Cesium is an OpenLayers-Cesium integration library to create the maps using OpenLayers and visualizing it on the globe using Cesium. Cesium is an open source JavaScript library for geospatial 3D mapping. OpenLayers alone can be used for just 2D mapping, but with its integration with Cesium can fulfill the needs of both 2D but also 3D mapping of raster and vector data sources. This 3D visualization provides the driver in the vehicle more interactive and helpful navigation towards his destination. Following figure shows an example of tilted/3D-map visualization using Ol-Cesium integration library.



Fig. 4.3: 3D tracking example using Ol-Cesium.js [26]

Ol-Cesium also supports map context (bounding box and zoom level), map selection and animated transitions between map and globe view [25].

**Nominatim**

Nominatim is a search engine to search OSM data by name or address (forward search) which generates synthetics coordinates. This process of forward search is called geocoding. Nominatim also provides functionality of looking up data by its geographic coordinates and generating associated addresses is called reverse search or reverse geocoding [17]. Nominatim will be used to implement one of key functionalities of our Location Software/Engine i.e. geocoding and reverse geocoding.

HERE Search REST API

**Geoserver**

Geoserver is an open source server for sharing geospatial data. It is designed for interoperability, it publishes data from any major spatial data source using open standards. Geoserver is an Open Geospatial Consortium (OGC) compliant product and can implement a number of open standards e.g. Web Map Service (WMS), Web Feature Service (WFS), Web Coverage Service (WCS) etc. [27]. Because of its high interoperability, it can work with many spatial data sources as shown in the Fig. 4.4:



Fig. 4.4: Geoserver interoperability [28]

In our Location Software, Geoserver will be used as middleware for rendering Geodata from backend to front end.

**PostGIS, pgRouting**

PostGIS is an open source software and it is an extension of PostgreSQL object relational database. It adds support for geographical objects to the PostgreSQL, hence allows for spatial/location SQL queries. PostGIS has feature of simple location awareness and many other features which can be hardly found in other competing SQL spatial databases such Oracle Spatial, Microsoft’s SQL Server [29]. PostGIS will be backend or spatial data storage of our Location Software which will have all the required databases such as road network database, POI database and database for dynamic data storage.

To work with road network database to add routing and navigation functionality in our Location Engine, there exists an extension of PostGIS/PostgreSQL geospatial database called pgRouting, which provides geospatial routing and other network analysis functionality. PostGIS and pgRouting will be used to experiment the dynamic routing feature and will not be used as part of the application which will run over Smart Core.

**Smart Core Studio**

It is software with a graphical UI to create configuration required for management of Smart Core resources to be used by operating systems of different guests.

“*Smart Core studio is designed for engineers that intend to work with Visteon hypervisor toolkit. Generally, Smart Core studio is used for constructing a hypervisor configuration essential for the management of guest Operating Systems. Device tree specification files describing the structure of this platform serve as input parameters for this tool. In addition, Smart Core Studio provides a list of useful features and visual artifacts to interact with dts files in order to configure its structure. Afterwards, it is expected that engineer compiles the project, generates binaries and flashes them to the board externally connected to the machine”* [30].

**Android Studio and SDK**

It is the official integrated development environment (IDE) for Google’s Android operating system, which is specifically designed for Android software development. Android studio provides fastest tools for developing applications which can run on every type of Android device. In this study the performance evaluation and assessment will be done for two types of front-ends or clients. One of them is Android based and the other one is Linux based. Both of these operating systems are supported by target hardware i.e. Smart Core

**GDAL (OGR2OGR)**

Geospatial Data Abstraction Library (GDAL) is an open source C++ conversion or translation library for raster and vector geospatial data formats. It is released by Open Source Geospatial Foundation (OSGeo) under open source X/MIT license. The GDAL library provides *single vector and raster abstract data models* to the calling application for all the supported formats. The library also comes with a variety of useful command line utilities for data translation and processing. Conventionally GDAL is used to design the raster part of the library and OGR deals with vector part [32]. GDAL also provides generic interface in a set of language bindings e.g. CSharp, Java, Perl and Python.

In this study a command line tool called ogr2ogr provided by the GDAL is being used for converting geospatial data into different formats, while providing customization options to reproject coordinates, minimize attributes, and a whole host of advanced options.

**GeoJSON Vector Tiles (geojson-vt)**

Geojson-vt is a highly efficient JavaScript library for slicing GeoJSON data into vector tiles on the fly, principally aimed to enable rendering and interacting with large geospatial datasets on the browser side without any server. Geojson-vt is created to power GeoJSON in Mapbox GL JS, but can be useful in other visualization platforms like OpenLayers, Leaflet and d3, as well as Node.js server applications.

The tiles resulting from slicing of GeoJSON data using geojson-vt conform to the JSON equivalent of the vector tile specification. In order to make data rendering and interaction fast, the tiles are simplified, retaining the minimum level of detail appropriate for each zoom level (simplifying shapes, filtering out tiny polygons and polylines) [33].

**Cordova/Phone Gap**

**YOURS Navigation API**

**HERE Geocoding API**

**HERE Places API**

* + - 1. **Architecture & Design of Software**

As mentioned in objectives of the study, the LBS software will have two different architectures i.e. for online and offline modes. The LBS software in online mode will have a three-tier architecture having client layer (Front-end), server layer (Middleware) and data layer (Backend). The Figure. 4.5 shows the three-tier architecture of the Location software. This three tier architecture will also be used to support dynamic routing feature in the online application, while the two-tier architecture which has no Geoserver as middleware is to create an application which will be standalone and will run on embedded system or Smart Core both in offline and online modes.

Map view

Other UI Componentss

Geo-server

Publish Geospatial Data

Receive/Respond requests from Client

Spatial Database/Backend

Road Networks Database + Online Basemap data hosting

POIs Database

Dynamic data storage



Client (Infotainment)



Interact with the Backend

Fig. 4.5: Location-Software Architecture

*Client side layer* includes the interface to be represented as an infotainment in vehicle for the driver. It would be built with JavaFx having web view for the map interface integrated in it. Since the mapping front end is built with HTML, JavaScript (OpenLayers) so it would give a web-based map interface. But the requirement of the application in the guest (infotainment) is that it should be native application which can be packaged and installed on the target platform, that’s why Cordova framework for android will be used for making the application native, and web-based mapping view will be integrated into it. The reason for using these web-based mapping technologies is that they are very lightweight and high-performance and applications built using them are rich cross-platform.

*Middleware layer* is the server layer which contains web server and Geoserver to receive and responding to requests from the client side, as well as connecting to the database for further operations etc. The business logic of the application also resides in the middleware.

*Backend layer* or *data layer* contains the data sources i.e. PostGIS database for storage and management of spatial data e.g. road networks, POIs, dynamic data etc. The Backend or data layer can also be any remotely hosted data sources such as Openmaptiles on the internet which implicitly contains road networks and other buildings as base data layer.

The above three-tier architecture is to add dynamic routing feature in LBS software using pgRouting, Geoserver and OpenLayers in a customized way and because of the hardware limitations of Smart Core, the Geoserver and PostGIS cannot be locally connected to our LBS application running on Smart Core and fetch the dynamic routing feature. This issue gives rise to usage of two tier architecture which can fetch all the required services (routing, geocoding, POI search) as well as backend data e.g. vector tiled Basemap data from internet. This two-tier architecture for online application can be seen below:

Map view

Other UI Componentss

Data Layer

Openmaptiles-Vector Tiles of OSM data as Basemap

HERE Search API for POI Search

HERE Geocoding API to search address for routing



Client (Infotainment)



The two-tier architecture for Offline Application differs from the online application by having standalone native data or backend layer in the LBS application to support offline maps which will be built using device’s local map data storage. The spatial data will be stored in GeoJSON file format and will be rendered as Vector Tiles using OpenLayers. Geojson-vt will be used to assist in rendering this large geospatial data on the fly by slicing it into vector tiles. The reason for using vector tiles is that they are lightweight and can be rendered on client side using local device’s hardware acceleration or GPU. This would enable us to better analyze the performance of our device or platform i.e. Smart Core.

**OpenLayers**

* style and visualize vector tiles to create a basemap layer
* visualize simulated route layer and user’s geolocation layer
* Animate/move the user’s position along route for navigation

**GeoJSON-VT**

to slice the GeoJSON data into Vector Tiles

**GeoJSON Files**

OSM Points.geojson

OSM Lines.geojson

OSM Polygons.geojson

### Deployment on Smart Core (Hardware)

The third component in this LBS platform is the computing platform or the hardware called Smart Core developed by Visteon. It is a multicore chip used as cockpit domain controller solution which aims at improving efficiency and cost of ownership [16]. The Fig. 4.6 shows that how Smart Core can be used to consolidate two different Electronic Control Units (ECU) operating on different operating systems, on a single SOC by means of virtualization:

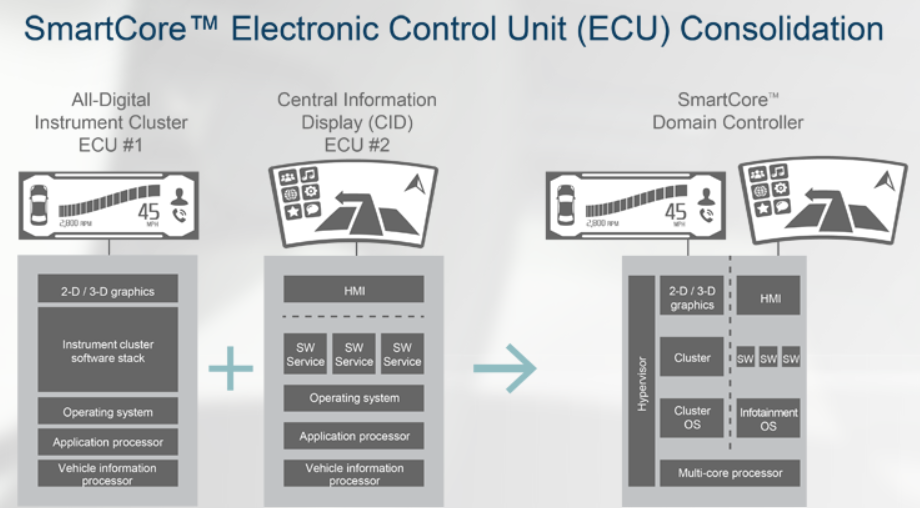


Fig. 4.6: Smart Core Electronic Control Unit (ECU) Consolidation [16]

* + - 1. Performance Analysis & Evaluation

The LBS application will be packaged and installed on Smart Core and then its performance on this multi core SOC will be evaluated and analyzed. Since the Smart Core is a single system on chip being used to operate multiple guests i.e. Cluster, Infotainment and Telematics, so the LBS application has to have a highly optimized performance on it. There are a few benchmarking parameters based upon which, the performance can be tested.

* + - * 1. Benchmarking

In benchmarking, a few tests will be done to measure the performance of the LBS application over the Smart Core. Following are some comparison criteria to evaluate the performance:

* Linux vs Android platform
* Raster vs Vector geo-data rendering comparison and evaluation
* Front end Mapping APIs comparison e.g. OpenLayers (Cesium) vs Leaflet
* Offline mode vs Online Mode

The application’s performance can be tested in both Linux and Android and can be compared. Similarly, the rendering of the geospatial data in two different formats i.e. vector and raster can be compared. The performance comparison of front-end mapping APIs can also be compared.

* + - 1. Performance Optimization

After the performance of LBS application over Smart Core has been evaluated, the next step will be to bring it to enhance it by choosing the best possible criteria and methods, in order to bring it to an optimized level.

# IMPLEMENTATION USING CASE STUDY

… (Implementation of above solution concept will be done later and complete development process will be presented in this section using the case study)

This chapter discusses the implementation of above solution approach for developing the LBS applications based upon the above given system architectures. The chapter begins with the development of LBS application with three-tier architecture which has Geoserver as middleware and will be used to implement the dynamic routing feature using PostGIS/pgRouting and with some sample data of obstacles or traffic jams to dynamically avoid and re-route. The second part in the implementation phase is the development of online LBS application with two-tier architecture which will be deployed on Smart Core and its performance will be analyzed. This

* + 1. **Development Process & Work Flow**

*Installing Software and Utilities*

* PostgreSQL
* PostGIS
* PgRouting
* Android Studio and Android SDK

*Downloading and import OSM data*

The first step after installing and configuring the required software and utilities is to setup an

* Download Baden Württemberg’s OSM data
* Import data into PostGIS using osm2pgsql tool

*Setting up Routing database*

* Enable pgRouting
* Perform routing tests and display the results in any desktop software like Quantum GIS (QGIS)
* Write SQL query for static and dynamic routing in order to avoid obstacles, traffic congestions etc.

*Setting up POI database*

* Extract all the required POIs from OSM points table in PostGIS database and setup a separate table/database
* Write and test SQL queries for POI database

*Setting Dynamic database*

* Add sample data in dynamic database
* Use this data for the dynamic routing query

*Setting Up Online Vector Tiles Basemap, Geocoding and Routing Services*

* Setup Openmaptiles service for Vector Tiles Hosting and Configure it with OpenLayers
* Register and Setup for HERE maps REST API service for Geocoding
* Register and Setup for YOUR Navigation REST API service for routing

*Preparing specific file database for offline native maps*

* Create GeoJSON files from OSM data using ogr2ogr

GeoJSON being a lightweight format for encoding a variety of geographic data structures, is used as Geodata format for our offline application. To be able to use GeoJSON based data, the point, line and polygon GeoJSON files from OSM data has to be prepared. This requires some setup using the command line tool called ogr2ogr from GDAL library. Ogr2ogr is a command line tool within the Geospatial Data Abstraction Library (GDAL), for converting geospatial data into different formats, while providing customization options to re-project coordinates, minimize attributes, and a whole host of advanced options.

For test purposes, the Baden Württemberg’s data source used in this study was reduced to Karlsruhe city. When running ogr2ogr, we have to state the type of data we're transforming to, the output file we want our transformed data to be saved into, and the OSM data file that has all the data, as shown below:

ogr2ogr -f GeoJSON Karlsruhe\_map.geojson karlsruhe-regbez-latest.osm

Executing the above command gives following error that:

ERROR 6: GeoJSON driver doesn't support creating more than one layer

The data we're transforming into GeoJSON, doesn't automatically create different sets of data that contain the various OSM layers. The individual layers have to be specified. To view the layers that are available, ogrinfo has to be executed on our OSM file. This will gives something like the following showing the layers:

Had to open data source read-only.

INFO: Open of `map.osm'

using driver `OSM' successful.

1: points (Point)

2: lines (Line String)

3: multilinestrings (Multi Line String)

4: multipolygons (Multi Polygon)

5: other\_relations (Geometry Collection)

So to be able to transform OSM data layers to GeoJSON, the specific layers also has to be specified. The ogr2ogr tool automatically will convert the spatial coordinates from OSM into CRS 84, the standard used for GeoJSON. To transform the points of interest POIs from the OSM map to GeoJSON, the ogr2ogr command has to be executed as shown below:

ogr2ogr -f GeoJSON Karlsruhe\_map.geojson karlsruhe-regbez-latest.osm points

This will create a GeoJSON data layer of all the points in the provided OSM file. Similarly the lines, polylines and polygon data layers in GeoJSON format are also generated by executing ogr2ogr commands on the same OSM file.

* Setting up Middleware (Geoserver)
* Configure Geoserver for PostGIS
* Connect with PostGIS data store
* Setup and style basemap from OSM data stored in PostGIS
* Serve the basemap both as vector and raster tiles
* Create SQL views for the above mentioned database queries i.e. routing queries, POI queries, and INSERT queries for asynchronous dynamic data insertion from client side to the dynamic data storage

*Development of Front-end Offline (Android Client: HTML + JavaScript (OpenLayers) + CSS)*

* Write the map view or interface using Ol-Cesium
  + Add the basemap layer in OpenLayers using geojson-vt integration library
  + Add the routing layer
  + Add the POI layer
* Create other UI utilities for interactive navigation and map exploration etc.
* Integrate the above web-based map view into JavaFx Scene or interface

*Development of Front-end Online (Android Client: Cordova (HTML + JavaScript (OpenLayers) + CSS))*

* Write the map view or interface using Ol-Cesium
  + Add the basemap layer
  + Add the routing layer
  + Add the POI layer
* Integrate the above web-based map view into other Android interface using Android web view component.

*Flashing the Software on Smart Core*

* Package, flash and install the software on Smart Core using Android Debug Interface (adb) command line tool
* Test the performance using the benchmarks

Define Use Cases

Run Profiling tool for every use case

Test and analyze the usage of system resources.

# 6. REFERENCES

[1] Muhammad Usman et al., “*Technologies and Solutions for Location-Based Services in Smart Cities: Past, Present, and Future*” (02.2018)

[2] TomTom, “Connected Navigation System”, <https://www.tomtom.com/automotive/automotive-solutions/connected-navigation-system> [Accessed: 08.2018]

[3] Jochen Schiller, Agnès Voisard, “*Location-Based Services*” (2003)

[4] Geolocation, <https://www.kisspng.com/free/vehicle-tracking-system-4.html> [Accessed: 06-08-2018]

[5] Karl Neumann, Sarah Tauscher - Spatial Databases and GIS – Ifis – TU Braunschweig, <http://www.ifis.cs.tu-bs.de/sites/default/files/course/wsem1213gischap10_pdf_17751.pdf>

[6] Kraak, M. J. & MacEachren, A. M., (2005), “Geovisualization and GIScience. Cartography and Geographic Information Science”, 32 (2): Page 67-68.

[7] ITWissen.info, Accessed – 07-08-2018 from - <https://www.itwissen.info/IVI-in-vehicle-infotainment-In-Vehicle-Infotainment.html>

[8] US Government Accountability Office, “IN-CAR LOCATION-BASED SERVICES” (09.2013), <https://www.gao.gov/products/GAO-14-81>

[9] Andy Graham et al., “Vehicle navigation systems – problems and potential”, Traffic Information Highway group (09.2006), Available on: <http://www.ciht.org.uk/en/document-summary/index.cfm/docid/EFD2C3D7-9CA1-4C5A-9D02C6FF3585EB18>

[10] HERE Routing API Documentation, <https://developer.here.com/documentation/routing/topics/introduction.html> [Accessed: 08.08.2018]

[11] Daniel Delling, Peter Sanders, Dominik Schultes, and Dorothea Wagner, “Engineering Route Planning Algorithms”, Universität Karlsruhe (TH) (2009), <http://ai.cs.unibas.ch/_files/teaching/fs15/ki/material/ki06-route-planning.pdf>

[12] TomTom, YouTube Video: “Peugeot's 3D Connected Navigation & TomTom services”, Available: <https://www.youtube.com/watch?v=lXh4F7H7ADE> [Accessed: 08.08.2018]

[13] Baden Württemberg “Our State”. <https://www.baden-wuerttemberg.de/en/our-state/the-state-and-its-people/> [Accessed: 13.08.2018]

[14] Geofabrik Downloads <https://download.geofabrik.de/europe/germany/baden-wuerttemberg.html> [Accessed: 13.08.2018]

[15] OpenStreetMap – Germany <https://www.openstreetmap.de/> [Accessed: 13.08.2018]

[16] Visteon, “Smart Core - Cockpit Domain Controllers”, <https://www.visteon.com/products/cockpit-domain-controllers.html> [Accessed: 20.08.2018]

[17] Nominatim, “Welcome to Nominatim”, <https://nominatim.openstreetmap.org> [Accessed: 21.08.2018]

[18] European Global Navigation Satellite Systems Agency, “What is GNSS”, <https://www.gsa.europa.eu/european-gnss/what-gnss> [Accessed on: 27.08.2018]s

[19] European Global Navigation Satellite Systems Agency, “GNSS Raw Measurements Task Force”, <https://www.gsa.europa.eu/gnss-raw-measurements-task-force> [Accessed on: 27.08.2018]

[20] Baden Württemberg Stiftung, “Mobiles Baden Württemberg”, <https://www.bwstiftung.de/fileadmin/Mediendatenbank_DE/BW_Stiftung/Programme_Dateien/Bildung/Nachhaltige_Entwicklung/Mobiles_Baden-Wuerttemberg/Zusammenfassung_Studie_Mobiles_BW_06.11.2017.pdf> [Accessed on: 28.08.2018]

[21] Baden Württemberg.de, “ Interaktive Baden-Württemberg-Karte - Unser Land”, <https://www.baden-wuerttemberg.de/de/unser-land/land-und-leute/interaktive-karten/interaktive-karte-des-landes-baden-wuerttemberg/> [Accessed on: 28.08.2018]

[22] Oracle, “What is JavaFx”, <https://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm> [Accessed on: 28.08.2018]

[23] Oracle, “JavaFx Architecture”, <https://docs.oracle.com/javafx/2/architecture/jfxpub-architecture.htm#A1107318> [Accessed on: 28.08.2018]

[24] OpenLayers, “Overview, Features”, <https://openlayers.org/> [Accessed on: 28.08.2018]

[25] Ol-Cesium, “OpenLayers-Cesium integration library – Features” <http://openlayers.org/ol-cesium/> [Accessed on: 29.08.2018]

[26] Ol-Cesium Examples, “3D tracking example”, <https://openlayers.org/ol-cesium/examples/tracking.html> [Accessed on: 29.08.2018]

[27] Geoserver, <http://geoserver.org/> [Accessed on: 29.08.2018]

[28] Boundless, “Geoserver”, <http://geoserver.org/> [Accessed on: 29.08.2018]

[29] PostGIS, “About”, <https://postgis.net/> [Accessed on: 29.08.2018]

[30] Visteon Smart Code Studio, “User manual” (2018)

[31] Ingrid C.M. Flinsenberg, Technische Universiteit Eindhoven. “Route Planning Algorithms for Car Navigation” (2004)

[32] “GDAL - Geospatial Data Abstraction Library” <https://www.gdal.org/index.html> [Accessed on: 19.11.2018]

[33] geojson-vt, “GeoJSON Vector Tiles”, <https://github.com/mapbox/geojson-vt> [Accessed on: 19.11.2018]