

Master's Thesis



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science and Engineering

Car infotainment application on a smartphone

Lukáš Hrubý

January 2016

Supervisor: Ing. Jan Šedivý, CSc.

Acknowledgement / Declaration

I would like to thank the supervisor of this thesis, Ing. Jan Šedivý, CSc., for the continuous support, for his motivation, enthusiasm, and advices he has given me over the past two years.

Beside my supervisor, my sincere thanks also goes to the members of Driving Simulation Research Group of Faculty of Transportation Sciences of Czech Technical University in Prague for all the help and patience with testing and everything related.

I would also like to thank my college, Bc. Michael Bláha for cooperating and advising throughout the whole thesis development process.

Last but not the least, I thank my family and friends for supporting me the whole time in my studies and life.

This work was partially supported by CZECH TECHNICAL UNIVERSITY MEDIA LABORATORY foundation.

I hereby declare that I worked out the presented thesis independently and I quoted all the sources used in this thesis in accord with Methodical instructions about ethical principles for writing academic thesis.

Prague, 11th January 2016

.....

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 11. ledna 2016

.....

Abstrakt / Abstract

Cílem této práce je navrhnout a implementovat aplikaci pro chytrý telefon, která je určena pro použití v automobilu a minimalizuje kognitivní zátěž. Aplikace je připojena k řídicí jednotce automobilu, zobrazuje a ukládá jízdní data. Dalším cílem je navržení způsobu, jak jsou jízdní data uložena v cloudovém prostředí a implementovat příklad. Posledním úkolem je vyhodnotit aplikaci pro chytrý telefon ve vhodném prostředí.

Klíčová slova: Android, Java, OBD, GUI, automobil, LCT, A/B testování, testování použitelnosti

The goal of this work is to design and develop an innovative smartphone application for in-car use with minimized cognitive load. The application is supposed to be connected to the vehicle's control unit, display and store the driving data. Next goal is to design a way, how the driving data are stored on a cloud and implement the example. The final goal is to evaluate the smartphone application using a suitable environment.

Keywords: Android, Java, OBD, GUI, car, LCT, A/B testing, usability testing


Contents /

1 Introduction	1	3 Design	19
1.1 Motivation	1	3.1 Requirements	19
1.2 Assignment analysis	2	3.2 Android Design Guidelines	19
1.2.1 Assignment tasks	2	3.3 Infotainment user interface	
1.3 Concept	3	architecture proposal	19
1.3.1 Types of User Interface	3	3.3.1 Portrait vs. Landscape	
1.3.2 What is Car User In-		mode	20
terface?	3	3.3.2 Use cases	21
1.3.3 Driving Distraction		3.4 Graphical User Interface	21
Factors	4	3.4.1 Low fidelity prototype ...	21
1.3.4 Automotive Human		3.4.2 High fidelity prototype ..	22
Factors	4	3.5 Logging server architecture	23
1.4 Examples of Car User Interface ..	5	3.5.1 Java Servlet API	23
2 Analysis	7	3.5.2 Mongo DB	23
2.1 Usage of portable devices	7	4 Implementation	25
2.2 In-car devices and controls	7	4.1 Development environments	25
2.3 In-car mobile applications	8	4.1.1 Android SDK	25
2.3.1 Android Auto	8	4.1.2 Java Platform, Stan-	
2.3.2 Apple CarPlay	9	dard Edition	25
2.3.3 Other vendors	9	4.1.3 Java Platform, Enter-	
2.4 Data & communication	10	prise Edition	26
2.4.1 In-vehicle real-time		4.2 Development tools	26
buses	10	4.2.1 Integrated develop-	
2.4.2 On-board Diagnostics		ment environment	26
(OBD)	11	4.2.2 Code version control	
2.4.3 Other types of commu-		system	26
nication	11	4.2.3 Continuous Integration	
2.5 Mobile platforms	12	Service	27
2.5.1 Android	12	4.2.4 Build Automation Sys-	
2.5.2 iOS	12	tem	27
2.5.3 Windows Phone (Win-		4.3 Third party libraries	27
dows 10 Mobile)	12	4.3.1 Mobile application	27
2.5.4 BlackBerry	12	4.3.2 Logging service	28
2.5.5 Firefox OS	12	4.4 Mobile Application Imple-	
2.5.6 Sailfish OS	12	mentation	28
2.5.7 Tizen	13	4.4.1 User interface for	
2.5.8 Ubuntu Phone OS	13	smartphones	28
2.6 Android mobile platform	13	4.4.2 Shared Core Library	29
2.6.1 Brief history of An-		4.4.3 Final application	29
droid operating system ..	13	4.5 Server implementation	30
2.7 Cloud computing	15	4.5.1 HTTP servlet	30
2.7.1 The NIST Definition of		4.5.2 Communication with	
Cloud Computing	15	server	31
2.8 Reference Hardware	16	5 Evaluation	33
2.8.1 Mobile device	16	5.1 Usability tests	33
2.8.2 OBD II devices	17	5.1.1 Used equipment	33

5.1.2	Participants	34
5.1.3	Tests schedule	36
5.2	Lane Change Test	37
5.2.1	Results	37
5.3	A/B test	38
5.3.1	Results	38
6	Conclusion	41
6.1	Assignment completion	41
6.1.1	Completing the assign- ment tasks	41
6.2	Summary	42
6.3	Future work	43
	References	45
A	Thesis assignment	49
B	List of Abbreviations	51
C	Usability test questionnaires	53
C.1	Screeners	53
C.2	Pre-test questionnaire	53
C.3	Post-test questionnaire	53
D	Usability test results analysis	55
D.1	MATLAB scripts	55
D.1.1	Statistics of A/B testing ..	55
D.1.2	Visualisation of Lane change test	56
D.2	Results of Lane Change Test ..	56
E	Content of the CD	59

Tables / Figures

1.1. Number of controls in cars.....5	1.1. Ford Model T dashboard6
2.1. Worldwide Devices Shipments by Device Type, 2014-2017 (Millions of Units)....7	1.2. Tatra T87 dashboard6
2.2. Worldwide Smartphone OS Market Share (Share in Unit Shipments) 12	1.3. BMW 3200CS dashboard6
5.1. Results of pre-test questionnaire 35	1.4. Skoda Favorit dashboard6
5.2. Results of post-test questionnaire 36	1.5. BMW X5 dashboard6
5.3. User testing schedule..... 36	1.6. Tesla Model S dashboard6
5.4. Results of LCT 37	2.1. Android Auto App running in Volkswagen Golf Mk. 69
5.5. Results of A/B testing 39	2.2. Screenshot of Google Maps in Android Auto App9
	2.3. Car dashboard with running Apple CarPlay 10
	2.4. LG Nexus 5..... 17
	2.5. MINI ELM327 bluetooth OBD2 V1.5 Bluetooth dongle . 18
	3.1. Limited information shown at display 20
	3.2. Low Fidelity mock up version 1 22
	3.3. Low Fidelity mock up version 2 22
	3.4. Servlet life – flow diagram 23
	4.1. Module displays single information. 29
	4.2. Module contains another set of child modules - folder. 29
	4.3. Swiping between different modules. 30
	4.4. Selection of Bluetooth device to connect to OBD dongle. 30
	4.5. Changing position of module using Drag and drop. 31
	4.6. Removing module using simple swipe gesture. 31
	5.1. Picture of simulator 34
	5.2. Eye tracking software 35
	5.3. Typical process of object avoidance during LCT 37
	5.4. Route of track used in A/B testing 38
	5.5. Visualization of typical speed and braking during A/B test .. 39
	6.1. Final CarDashboard mobile application with the day mode color scheme 42



6.2.	Final CarDashboard mobile modified for use in smart buildings	43
D.1.	Lane Change Test - Participant #1.....	57
D.2.	Lane Change Test - Participant #2.....	57
D.3.	Lane Change Test - Participant #3.....	57
D.4.	Lane Change Test - Participant #4.....	57
D.5.	Lane Change Test - Participant #5.....	57

Chapter 1

Introduction

Mobile devices have become an inseparable part of our lives. They tend to integrate more and more functions in one device. This approach has advantages and disadvantages. At the end, the described situation leads to the fact that people are using smartphones all the time. The need of current information from news or social networks is enormous. With the expansion of the wireless internet connection, more devices are connected to this global network.

Major problem reveals when people try to use the modern smartphones equipped with large touchscreens while driving a car. User interfaces of those devices are not designed for this purpose and that might cause hazard situation and even the road accidents.

This thesis will try to solve this approach by examination of a new way, how to design the mobile applications for in-car use.

1.1 Motivation

With a huge progress of mobile development in the last few years many interesting challenges emerged. Internet connection is available all around the world and the internet population was more than 3.2 billion by the end of 2015. Price of hardware is continuously decreasing, which leads to massive growth of the trend called Internet of Things. Gartner, Inc estimates that 1.6 billion connected things will be used by smart cities in 2016 (an increase of 39% from 2015) ¹⁾. This trend naturally comes also to the automotive environment. Modern cars in European Union are required to be connected to the mobile network due to the presence of eCall system²⁾. From this point, it is only a small step to have fully connected vehicles with the possibility of data sharing.

Each vehicle produces many information about itself. The data are in most cases not utilized, although they bring big opportunity in various use-cases. OEMs could avoid costly product recalls and use them in vehicle development. Dealerships could recognize and respond to customer needs. Information about current condition of vehicles could enable repair shops to order the necessary spare parts in advance to eliminate delays. And drivers could utilize these data to arrange service interval according to the real use of car and therefore save money [2].

The only problem is with the penetration of new technologies into current cars. Improvements in vehicle quality have made cars more reliable and customers are not forced to change the vehicles so often. In 2015 the average age of vehicles on the road in the U.S. was 11.5 years which is a record-high value [3].

¹⁾ <http://www.gartner.com/newsroom/id/3175418>

²⁾ eCall is an initiative with the purpose to bring rapid assistance to motorists involved in a collision anywhere in the European Union [1]

1.2 Assignment analysis

In this section the assignment will be analyzed. That will lead to precise definition what should be included in this thesis and what should not be included. The assignment contains the whole process of software development from analysis to design, development, and testing, which makes it a perfect topic for a diploma thesis in Software Engineering field.

1.2.1 Assignment tasks

The assignment of this thesis is divided into different tasks. From these tasks the structure of the thesis was designed. Each chapter refers to one important part of software development process.

1.2.1.1 Study the current trends in controlling car infotainment systems

This introductory part is about common usage of a infotainment systems in past, present, and future. It is important to show the progress that have been made in last few years.

1.2.1.2 Explore existing mobile applications for in-car use

Cars are becoming more and more connected with smartphones. People tend to use their smartphone while driving. Software producers are aware of this fact. This task should contain the most important competitors in this discipline. As smartphones are running various operating system, it is handy to involve the basic overview of currently used mobile operating systems.

1.2.1.3 Study communication with car ECU (CAN, OBD, etc.)

There are many types of communication in a car. One type of these communication ways is the communication with a car's ECU. Since it is important to know how utilize this communication, other communication types should be mentioned.

1.2.1.4 Design and implement mobile application for in-car user

From the knowledge gained in previous tasks, a logical next step is to design and implement a solution to discovered problems. This thesis focuses on development a mobile application that is aimed to be used while driving. Certain constraints specific for in-car usage must be taken in account.

1.2.1.5 Implement logging server

Big data and analytics technology formed the whole new science discipline called Data Science. Modern cars are producing much data about their usage and drivers' behaviour. This information can be utilized and used to help people to drive better. In this task, logging server for gathering and storing driving data is developed.

1.2.1.6 Integrate mobile application with logging server

As the amount of data logged during drives can be really big, there is a need to find a way how to transfer them from the mobile application to the logging server. This way should not affect common use of a smartphone and must take in account various limits in smartphones.

1.2.1.7 Compare your solution with existing solutions

The final application must be competitive in its field of use. Therefore it should be better in some ways. In this task, the application is compared with the one or more competitors that are already on the market.

■ 1.2.1.8 Evaluate the application using usability testing

Driving a car is a high cognitive load demanding task. Additional cognitive imposed to a drive could lead to dangerous situations on the road. The final application must be evaluate to prove its safe use while driving.

■ 1.3 Concept

■ 1.3.1 Types of User Interface

User interfaces can be divided into different categories according to various classifications. These classifications change with time and used technologies. For the purpose of this work, a short overview of UI types with relationship to in-vehicular use will be discussed [4].

■ 1.3.1.1 Hardware UI / Haptic UI

Hardware user interfaces are tangible and user is able to touch them and feel the feedback. Standard elements are e.g. switches, wheels, and LEDs. In vehicles this approach of UI is used since the beginning. In last decades the controls are adopting multiple purposes - for instance combined controls like push/turn control knobs. Newer versions include a touch pad to recognize gestures and letters. In mobile devices this approach was applied in previous generation of mobile devices equipped with hardware keyboard.

■ 1.3.1.2 Display-based UI

User interfaces enriched with LED or LCD devices can be seen from two different points of view. In the first one, the display is just an addition to previously mentioned Hardware User Interface. User interfaces of this type - graphical user interfaces (GUI) have been introduces in 1981 as part of the Xerox Star workstation, which used the WIMP (windows, icons, menus, and a pointing device) for the first time [5].

The second point of view is more important from the current perspective. Display is the only hardware part of UI and allows an interaction directly on the screen using a touch-screen layer. These systems are nowadays commonly used in mobile phones, tablets, laptops, and also in IVIS (in-vehicular interactive systems). The advantages are obvious - UI is more flexible and can be changed with different needs of a particular application. Even though touch-screen devices are used in recent models of cars, the last research shows that the distraction of drivers while using a touch-screen devices appears to be too high [6].

■ 1.3.1.3 Mobile Device UI

Modern mobile devices, so-called smartphones are equipped (beside large touch-screens) with wide range of sensors, which can be seen as a part of the user interface. These sensors can be used to enrich the user experience (e.g. changes of portrait and landscape modes).

■ 1.3.2 What is Car User Interface?

In the last few years it can be observed the trend of pervasive computing (or so called ubiquitous computing) is impacting everyone's life more than before. Microprocessors are being embedded in various type of devices and other everyday objects. These devices and objects can therefore exchange information about their status or environment. The

automotive industry is naturally part of this growing trend and recent cars contain hundreds of microprocessors. But with all the convenience that comes with the modern technology, there is also need for the way of controlling these devices.

The main difference between the conventional user interface (e.g. used in PCs) and pervasive user interfaces is the demand for users' limited attention (as they navigate the external world). Whereas interfaces on PCs are typically the primary focus of a person's attention (primary task), pervasive user interfaces are one of potentially many other tasks that a person must perform. This problem becomes critical when the usage of user interface is secondary task to some performance-critical primary task. In this case, the car controlling (speed, direction, etc.) is definitely the primary task and controlling in-car devices such as infotainment system is a secondary task. The design and evaluation of this kind of interfaces must take into account the impact of using it on the primary task (in this case – driving).

■ 1.3.3 Driving Distraction Factors

The human's cognitive load capacity is constant. Drivers with long experience have learned to handle some of the tasks almost automatically, for example gear shifting or rear view checking is not impacting their attention to driving [7]. On the other hand, the growing number of cars on the roads, more miles driven, constantly increasing number of road signs and large number of buttons in the cockpit enlarge the number of events that drivers have to handle. Not paying full attention to driving increases the probability of getting into an accident.

Driving and paying attention to the road situation is the main driver's task. All the other activities are the secondary tasks. The driving a car is a two-dimensional task, where the first dimension includes the maintenance of the direction – staying in the lane. Maintaining the safe distance of the followed car is the second dimension. The danger of leaving the correct lane or getting too close to the car ahead can be quantified by the time interval, in which the driver does not make a correction. This may be the time the driver is pointing his eyes to some of the secondary activities. There are many studies discussing the safe period of time to take the eyes from the road, but this is not the topic of this study. The simple conclusion for us is that the shorter is the eyes off road period due to controlling the in-car applications the safer they are. After all, the main goal is to make the driving safer, therefore generally speaking we will try to suggest solutions with lower distraction to the driver. There are many other factors influencing the safety, such as weather, light conditions etc. The task is to make the UI of applications less distractive. These facts will be respected in the UI design when appropriate, for example changing the screen color scheme between day and night [8].

■ 1.3.4 Automotive Human Factors

European Commission reported in the press release¹⁾ that number of traffic accidents is continuously decreasing but still there are 1.3 million fatalities and 50 million injuries p.a. worldwide. 28 thousands people die in the EU, another 40 thousands die in the USA. World Health Organization estimates that by 2020, traffic accidents will be the third highest cause of death²⁾. Road traffic accidents and injuries cause also considerable economic losses to victims, their families, and to the nations as a whole.

¹⁾ http://europa.eu/rapid/press-release_IP-13-236_en.htm

²⁾ <http://www.who.int/mediacentre/factsheets/fs358/en/>

Driving can be considered as completing certain goals. The main goal is to reach the destination. As the other goals can be considered *Productivity* (i.e. drive as fast as possible) and *Safety* (i.e. reduce personal risks - no injuries, crashes, etc.)

The important question for analysis is *What are the tasks driver is doing when driving?* There are three general categories of activities [9]:

- Strategic:
 - Purpose of a trip,
 - driver's general goals.
- Tactical:
 - Choice of maneuvers,
 - immediate goals, e.g., in getting to a destination,
 - speed and/or gear selection,
 - decision to pass,
 - choice of lanes.
- Control:
 - Moment-to-moment operation of vehicle,
 - maintaining desired speed,
 - keeping the desired distance from the car ahead,
 - keeping the car in the lane.

1.4 Examples of Car User Interface

In this section a few examples of user interfaces used in historical and modern cars will be introduced. As examples the pictures of dashboard of Ford Model T ¹⁾, Tatra T87 ²⁾, BMW 3200CS ³⁾, Skoda Favorit ⁴⁾, BMW X5 ⁵⁾, and Tesla Model S ⁶⁾ were used. Table 1.1 shows a simple comparison of number of controls in historical and modern cars.

Car type	Years of production	Amount of dashboard controls
Ford Model T	1908-1927	2
Tatra T87	1936-1950	c. 10
BMW 3200CS	1962-1965	c. 20
Skoda Favorit	1987-1995	c. 40
BMW X5	1999-present	c. 100
Tesla Model S	2012-present	unknown ⁷⁾

Table 1.1. Basic information about participants in usability testing.

¹⁾ <https://www.flickr.com/photos/15378728@N00/5021836954/in/photostream/>

²⁾ https://en.wikipedia.org/wiki/Tatra_87

³⁾ http://www.carsbase.com/photo/photo_full.php?id=39388

⁴⁾ <https://upload.wikimedia.org/wikipedia/commons/0/09/Skodafavoritinside.JPG>

⁵⁾ <http://www.seriouswheels.com/2011/klm/2011-Mansory-BMW-X5-Dashboard-1280x960.htm>

⁶⁾ <https://www.flickr.com/photos/gbpublic/10994279865>

⁷⁾ Number of controls in Tesla Model S car user interface is unknown as it varies with every application.

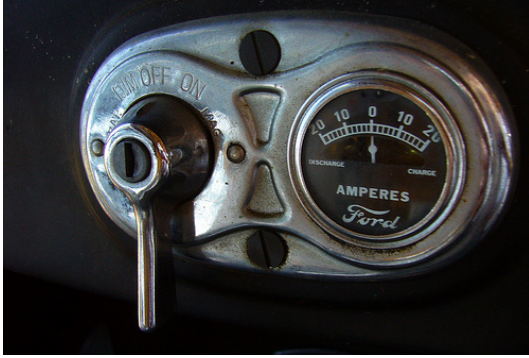


Figure 1.1. Ford Model T dashboard



Figure 1.2. Tatra T87 dashboard



Figure 1.3. BMW 3200CS dashboard



Figure 1.4. Skoda Favorit dashboard



Figure 1.5. BMW X5 dashboard



Figure 1.6. Tesla Model S dashboard

Chapter 2

Analysis

This chapter is dedicated to the analysis of various topics related to this thesis. In the first part, current trends in shipments of devices will be introduced. This introduction will be followed by analysis of the latest in-car infotainment solutions including their advantages and disadvantages. Then the basic facts about various types of communication in automotive environment will be explained. The last section of this chapter is devoted to cloud computing topic and its extension to in-vehicular use.

2.1 Usage of portable devices

The shipments of portable devices are facing big changes recently. Sales of traditional PCs are decreasing every year. On the other side, the shipments of portable devices (mainly mobile phones and so-called ultramobile devices) are experiencing the opposite trend. As we can see in table 2.1, the numbers are increasing. According to Gartner Inc. [10] the mobile phone shipments are by far the largest. These numbers suggest what kind of devices are customers use the most and consequently what devices will be most probably used while driving.

Device Type	2014	2015	2016	2017
Traditional PCs (Desk-Based and Notebook)	277	251	243	233
Ultramobiles ¹⁾ (Premium)	37	49	68	89
PC Market	314	300	311	322
Ultramobiles (Tablets and Clamshells) ²⁾	226	214	228	244
Computing Devices Market	540	514	539	566
Mobile Phones	1879	1940	2007	2062
Total Devices Market	2419	2454	2546	2628

Table 2.1. Worldwide Devices Shipments by Device Type, 2014-2017 (Millions of Units) [10]

2.2 In-car devices and controls

Driving a car is a stressful task. Distraction is the number one reason for traffic accidents. The reasons are obvious:

- Technological development is faster than the ability to handle it. Product life cycle (in automotive environment) is getting shorter - from more than 4 years to less than 18 months.
- Infotainment systems functionality is increasing, but the human processing capacity is the same.
- Complexity of whole eco system of human mobility is continuously increasing [11]:
 - High volume of traffic
 - More miles driven

- Complex road infrastructure
- Overusage of road signs

Hardware interfaces are a mixture of knobs, buttons, sliders, and switches in the car. These controls are called **Tangible User Interfaces** [12]. They are the traditional car on touch and physical environment. They allow the driver to control the car. In the last decades the mechanical buttons have changed to electrical switches with micro controllers. The measurement equipment was replaced with screens. The whole car infrastructure became digital. This change is following similar changes in the digital personal devices.

There are other devices in the car or devices the driver and passengers will bring to the car, which will require wired or wireless connection. First USB, for MP3 players, phone, or it can be used to charge some of the mobile devices while driving. Some cars may provide flash memory slot supporting different types and allowing music playback or allow replication to the in-car hard drive.

The car may also provide a Wi-Fi connection in several different setups. Other mobile devices the driver and the passengers bring to the car must be considered. The top level car may also provide HDMI for in-car screens located on the dashboard and back seats.

2.3 In-car mobile applications

In this section the important vendors of mobile applications aiming for in-car use will be introduced. The two most popular competitors - Android Auto and Apple CarPlay will be presented in details. Other vendors of applications available on various mobile application stores will be featured.

2.3.1 Android Auto

Android Auto is an app that integrates with a car to make it easier for driver to use some of the main features of the Android phone while driving. Driver can control things like navigation, music, and the phone's dialer from the car's digital display so that the driver can stay focused on the road. Android Auto app can be downloaded on Google Play¹). There is also a need for compatible car or aftermarket unit. Most Android phones running Android 5.0 Lollipop can run Android Auto in a compatible car or aftermarket unit. In addition to the phone itself, Android Auto requires a data connection and a Google Account²) to download the app from the Google Play Store [13].

Communication with the phone is ensured by USB cable which provides data transfer and recharging capabilities. In-dashboard system uses also the internet data connection from the phone. In figure 2.1 is shown the general concept of using the Android Auto ecosystem. In figure 2.2 is then shown the navigational feature using Google Maps.

Android Auto concept is heavily focused on minimizing the driver distraction. Therefore only three possibilities to interact with a driver are present:

- System overview (weather, time, navigation, etc.),
- audio applications (Google Play Music, Spotify, etc.),
- messaging application (Messenger, WhatsApp, etc.).

¹) <https://play.google.com/store?hl=en>

²) <https://accounts.google.com/SignUp>



Figure 2.1. Android Auto App running in Volkswagen Golf Mk. 6

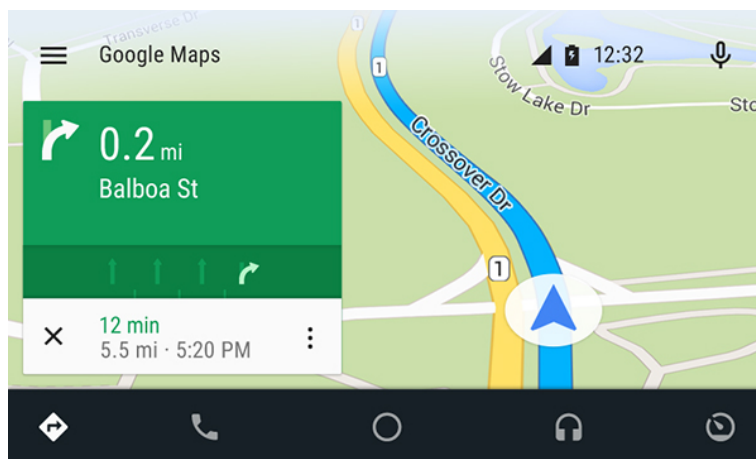


Figure 2.2. Screenshot of Google Maps in Android Auto App

■ 2.3.2 Apple CarPlay

Apple CarPlay is the main competitor of Android Auto. It offers a way to use Apple iPhone smartphones in vehicles. An example of integration this system to car is in figure 2.3. Key features of Apple CarPlay include the following [14]:

- Maps
- Messages
- Phone
- Music
- 3rd Party Apps

■ 2.3.3 Other vendors

Android Auto and Apple CarPlay are not the only available solutions of infotainment systems. Almost every car manufacturer develops its own solution of IVIS, e.g. BMW



Figure 2.3. Car dashboard with running Apple CarPlay

ConnectedDrive¹), Ford Sync²) or General Motors' OnStar³). Apple's and Google's solutions were chosen because they both offer an open platform for 3rd party developers to publish their applications to well-known application stores.

2.4 Data & communication

2.4.1 In-vehicle real-time buses

Controller Area Network Bus (CAN-BUS) - Bosch originally developed the Controller Area Network (CAN) in 1985 for in-vehicle networks. In the past, automotive manufacturers connected electronic devices in vehicles using point-to-point wiring systems. As more and more electronics started to be used in vehicles, the situation led to very complex and confusional systems that were heavy and expensive. They then replaced dedicated wiring with in-vehicle networks, which reduced wiring cost, complexity, and weight. CAN is a high-integrity serial bus system for networking intelligent devices, developed as the standard in-vehicle network. Whole automotive industry quickly adopted CAN and it became the international standard in 1993 known as ISO 11898 [15].

FlexRay - The FlexRay communication bus is a deterministic, fault-tolerant, and high-speed bus system developed in conjunction with automobile manufacturers and leading suppliers. FlexRay delivers the error tolerance and time-determinism performance requirements for x-by-wire applications (i.e. drive-by-wire, steer-by-wire, brake-by-wire, etc.). This bus was developed to provide features that CAN cannot offer (e.g. synchronization or high performance) [16].

¹) www.bmw.es/connecteddrive

²) www.ford.com/technology/sync

³) <https://www.onstar.com/us/en/home.html>

■ 2.4.2 On-board Diagnostics (OBD)

The first generation of On-Board Diagnostic requirements, called OBD I, was developed by the California Air Resources Board (ARB) and implemented in 1988. As technological possibilities to expand On-Board Diagnostic capability increased, a second-generation of On-Board Diagnostics requirements was developed. This second version is called OBDII. The Clean Air Act Amendments of 1990¹⁾ stated that, beginning with the 1996 model year, all light-duty vehicle and trucks made available for sale outside of the state of California must also be equipped with OBDII. In addition, EPA also requires that medium duty vehicles up to 6.5 tons must also be equipped with OBDII systems beginning in the 2004 model year [17]. In the last years, OBD systems are required by EPA to be installed on light-duty vehicles and trucks, as well as heavy-duty engines.

OBD plays an important role where vehicle inspection and maintenance programs are required. OBD serves as an early warning system that alerts to the potential need for vehicle repair through the *Check Engine* light on the dashboard of a vehicle [17].

■ 2.4.3 Other types of communication

As the application supports to communication with a car's ECU, it connects to the internet and uploads data logs, various types of communication are taken in account.

■ 2.4.3.1 Bluetooth

Bluetooth is wireless technology standard for communication over short distances. It has been used mainly to connect headsets to phones. Nowadays it is gaining more popularity in the Internet of Things.

Support of Bluetooth technology is crucial for the final application. Bluetooth is used for connection to ELM327 Bluetooth Dongle which provides communication with OBD interface. Dongles based on ELM327 usually supports USB, RM-232, WiFi or Bluetooth connections. In this case the Bluetooth connection was chosen because of simplicity of usage, commercial availability of the dongles, and low power consumption²⁾.

■ 2.4.3.2 Mobile data communication (3G, 4G)

Category of mobile telecommunication in this case includes the possibilities of connection to the internet when wi-fi connection is not available. To this category belong wireless network technologies provided by mobile carriers (3G, 4G). This type of internet connectivity is not supposed to be used for uploading trip logs due to limited data plans issue. The typical use is to control other devices via HTTP request initiated from the smartphone (e.g., opening garage doors or entrance gates). The application is, however, usable without the data plan.

■ 2.4.3.3 Wi-fi

Although the Bluetooth connection seems to be an ideal way how to connect to a car's ECU, there are devices on the market that support also Wi-Fi connection. The advantage is a bigger bandwidth provided by Wi-Fi (proposed versions Wi-Fi Direct promises 250 Mbps while Bluetooth 4.0 promises 25 Mbps³⁾). This bandwidth can be used to transfer more data from the ECU. However, for the purposes of the final application this difference is not significant, therefore only Bluetooth support is proposed.

¹⁾ <http://www.epa.gov/clean-air-act-overview/1990-clean-air-act-amendment-summary>

²⁾ <http://science.opposingviews.com/bluetooth-vs-wifi-power-consumption-17630.html>

³⁾ http://www.pcworld.com/article/208778/Wi-Fi_Direct_vs_Bluetooth_4_0_A_Battle_for_Supremacy.html

2.5 Mobile platforms

Nowadays, mobile devices are shipped with various operating systems. Each OS has advantages and disadvantages. This section introduces the most important mobile operating system's vendors. Market share of most significant operating systems in in table 2.2.

Period	Android	iOS	WP	BlackBerry OS	Others
2015Q2	82.8%	13.9%	2.6%	0.3%	0.4%
2014Q2	84.8%	11.6%	2.5%	0.5%	0.7%
2013Q2	79.8%	12.9%	3.4%	2.8%	1.2%
2012Q2	69.3%	16.6%	3.1%	4.9%	6.1%

Table 2.2. Worldwide Smartphone OS Market Share (Share in Unit Shipments)¹)

2.5.1 Android

Android is an open source from Google Inc. As it is the most used mobile operating system, it was chosen as the target platform for the purposes of this thesis and it is described in detail in further sections [18].

2.5.2 iOS

iOS (also known as iPhone OS) is being developed by Apple Inc. [19] It is closed sourced and proprietary and is supposed to run only on devices produced by Apple Inc - Apple iPhone, iPod Touch, iPad, and Apple TV. It allows installing native third party applications.

2.5.3 Windows Phone (Windows 10 Mobile)

Windows Phone is a closed sourced and proprietary mobile operating system from Microsoft [20]. It offers rich integration with other serviced provided by Microsoft such as OneDrive, Xbox Music & Video, Bing, and Office. From January 2015 it is re-branded as Windows 10 Phone.

2.5.4 BlackBerry

BlackBerry (currently version 10) was one of the dominant platform in mobile world once. It is a closed source and proprietary operating system and runs only on devices produced by BlackBerry. By the end of 2014 the market shares were less than half of percent [21].

2.5.5 Firefox OS

Firefox OS is from Mozilla [22]. It is released under the Mozilla Public License and uses kernel and drivers from Android. Firefox OS was publicly demonstrated in February 2012.

2.5.6 Sailfish OS

Sailfish OS is from finnish company Jolla [23]. This operating system (and company) was established by former employees of Nokia and its abandoned operating system MeeGo. Sailfish OS is partly open sourced under GPL license.

■ 2.5.7 Tizen

Linux based operating system Tizen is developed by Linux Foundation [24], Tizen Association, Samsung, and Intel. It is based on Linux kernel and C library implementing Linux API. It is not only mobile operating system but also targets wide range of devices including smart TVs, smart cameras, wearables, in-vehicle infotainment devices, and printers. It is open source software.

■ 2.5.8 Ubuntu Phone OS

Ubuntu Phone is open source OS from Canonical Ltd. [25] It is a mobile version of desktop OS produced by the same company. It uses kernel and drivers from Android similarly to Firefox OS.

■ 2.6 Android mobile platform

Android is a software stack for mobile devices that includes an operating system, middleware, and key applications. The various components of Android are designed as a stack, with the ‘Applications’ forming the top layer of the stack, while the Linux kernel forms the lowest layer. Android ships with a set of core applications including an e-mail client, SMS program, calendar, maps, browser, contacts, and other features. All applications are written using the Java programming language.

Developers have full access to the same framework APIs used by the core applications. The application architecture is designed to simplify the reuse of components; the capabilities of any application can be published and then be made use of by any other application (subject to security constraints enforced by the framework). The same mechanism allows components to be replaced by the user.

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language. Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format, which is optimized for minimal memory footprint. The VM is register-based and runs classes compiled by a Java language compiler that has been transformed into the .dex format by the included dx tool. The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management [26].

■ 2.6.1 Brief history of Android operating system

In this subsection a brief history of Android platform [18] is presented. Knowing the history of the operating system is important to understand further parts of this work.

■ 2.6.1.1 Android 1.0

The very first version of Android was released in October 22nd, 2008 when the first smartphone - T-Mobile G1 launched in the United States. This version lacked functionality such as on-screen keyboard (T-Mobile G1 was equipped with hardware keyboard) or multi touch capability. On the other side, many functions that are important even in latest version were introduced: Gmail integration, Android Market or Pull-down notification window.

■ 2.6.1.2 Android 1.1

The purpose of this version was to show that update of the smartphone's operating system can be quick and painless. There was no need of flashing new firmware by special tools or delete all data. Everything went seamlessly over the internet.

■ 2.6.1.3 Android 1.5 Cupcake

Released in April 2009, half a year after first G1s were shipped, this version added on-screen keyboard, clipboard (copy & paste) support, and video capture and playback. This version also started the 'sweet' naming convention when every version of operating system is named after some dessert starting with letters in alphabetical order. The classical Google's search box was included as well.

■ 2.6.1.4 Android 1.6 Donut

Android 1.6 added support for CDMA networks which led to increased sales in US and Asia where mobile carriers were running these types of mobile networks. The support for different resolutions was added: QVGA, HVGA, WVGA, FWVGA, qHD, and 720p.

■ 2.6.1.5 Android 2.0 / 2.1 Eclair

Released one year after the Android's premiere, this version offered support for multiple Google accounts, Google Maps Navigation, speech to text, and new swiping unlock screen. With the Android 2.1, the new stock-Android phone was introduced - Google Nexus One.

■ 2.6.1.6 Android 2.2 Froyo

Froyo confirmed the position of Nexus phones as Google Nexus One got the upgrade first. This version also added support for mobile hot-spot which was immediately disabled by many carriers.

■ 2.6.1.7 Android 2.3 Gingerbread

Half a year after Froyo, Android 2.3. Gingerbread was released. Beside improvements of keyboard or copy & paste, Google started to care about battery life. Poorly developed applications had been able to drain phone's battery in very short time. Gingerbread brought tools to monitor battery usage. Another important feature that came with this version was support for front facing cameras and NFC.

■ 2.6.1.8 Android 3 Honeycomb

This version was not aimed for usage on phones. The main target were tablets. Google switched from green colors palette to blue one. Honeycomb previewed an important redesign of whole platform's user interface.

■ 2.6.1.9 Android 4.0 Ice Cream Sandwich

Android 4.0 was without any doubts the biggest change for the Android ecosystem so far. Majority of graphical elements was redesigned, new bespoke font called Roboto came.

■ 2.6.1.10 Android 4.1 Jelly Bean

Jelly Bean firstly came with the application Google Now which is advanced predictive and question-answering system present in whole Google's platforms. Minor design changes were performed and system applications were improved. Google also started to advertise the Play Store Edition of selected smartphones that offered stock Android running on phones by various manufacturers. This approach was not successful and later Google returned back to Nexus series.

■ 2.6.1.11 Android 4.4 KitKat

The main aim for this version was to make Android faster, more efficient, and less resource intensive. The Nexus 5 smartphone was firstly released with this version.

■ 2.6.1.12 Android 5 Lollipop

Fistly named just Android L, this version came with so called Material Design. Google wanted to unify user interfaces of 3rd party application and provide guidelines for developers. Android Wear and Android Auto concepts were introduced.

■ 2.6.1.13 Android 6 Marshmallow

In the time of publishing this work. Android 6 is the latest version. It concerns about energy consumption and security of users. Marshmallow allows applications to ask for permission to some resource while using the application. Before that permissions were asked only during installation process.

■ 2.7 Cloud computing

The smartphones connected to the internet is the key in the car ecosystem. All smartphone allow access to traditional desktop applications such as PIM managers. Most of the downloadable applications are using large databases to store the user's data and profiles.

For the purposes of this thesis, cloud computing is used as environment to host the logging server. It offers easy scalability and coverage of different geographical locations as the cloud providers have their datacenters in different part of the world.

■ 2.7.1 The NIST Definition of Cloud Computing

National Institute of Standards and Technology (NIST) defined the cloud model as composition of five essential characteristics, three service models, and four deployment models [27].

■ 2.7.1.1 Essential Characteristics

- **On-demand self-service** - A consumer can change capabilities (processor time, memory, storage etc.) automatically without requiring human interaction,
- **broad network access** - Capabilities are available over the network using standard mechanisms,
- **resource pooling** - Provider's computing resources are pooled to serve multiple consumers. These resources are location independent,
- **rapid elasticity** - Capabilities can be elastically provisioned and released. Capabilities available for provisioning often appear to be unlimited from the consumer's point of view,
- **measured service** - System controls a optimizes resource usage. The usage of resources is monitored, controlled, and reported to provide transparency for provider and consumer of the utilized service.

■ 2.7.1.2 Service Models

- **Software as a Service (SaaS)** - Consumer uses provider's applications running on a cloud infrastructure. Applications are accessible from various client devices (e.g., web-browser, third party application or API). Consumer does not manage or control

underlying cloud infrastructure neither individual application capabilities but can control user specific application configuration,

- **Platform as a Service (PaaS)** - Consumer has a capability to deploy onto the cloud custom applications created using the environment supported by the provider. Consumer does not manage or control underlying cloud infrastructure but has control over the deployed applications and can configure settings for the application environment,
- **Infrastructure as a Service (IaaS)** - Consumer provisions directly processing, storage, networks, and other fundamental computing resources where is he able to deploy and run arbitrary software (including operating systems). Customer does not manage the underlying infrastructure but has control over OSs, storage, and deployed applications.

■ 2.7.1.3 Deployment Models

- **Private cloud** - The cloud infrastructure is provisioned for single organization use. It can be owned directly by the organization or provided by a third party,
- **community cloud** - The cloud infrastructure is provisioned for the specific single community of consumers. It can be owned by one or more community's member or a third party,
- **public cloud** - The cloud infrastructure is provided for open use by the general public. It exists on the premises of the cloud provider,
- **hybrid cloud** - The cloud infrastructure is a composition of two or more previously mentioned models that are bound together by standardized or proprietary technology to enable data and application portability.

■ 2.8 Reference Hardware

■ 2.8.1 Mobile device

Mobile application is supposed to run on wide range of devices available in current market. It does not require any hardware or software customization of target device. It requires Android 5.0 and higher. This version of Android OS is available for majority of current shipped devices.

■ 2.8.1.1 LG Nexus 5

LG Nexus 5 was used for the purposes of development and testing. The main reason for selection was the fact, that whole life cycle (design, development, marketing, and support) is managed directly by Google, Inc, the vendor of Android OS. Even though the development was aimed to reach the full functionality of as wide range of devices as possible, there can still be minor issues on other devices.

The technical specification of the device are following [28]:

- **Codename:** Hammerhead
- **Developer:** Google, LG Electronics
- **Manufacturer:** LG Electronics
- **Compatible networks:** 2G/3G/4G LTE
- **Dimensions:** 137.84 mm H; 69.17 mm W; 8.59 mm D
- **Weight:** 130 g
- **Operating system:** Android 'Marshmallow' 6.0.1

- **System on chip:** Qualcomm Snapdragon 800
- **CPU:** 2.26 GHz quad-core Krait 400
- **GPU:** Ardeno 330, 450 MHz
- **Memory:** 2 GB of LPDDR3-1600 RAM
- **Storage:** 16 GB (12.55 GB available)
- **Display:** 4.95 in (126 mm) Full HD 1920x1080 px (445 ppi) IPS LCD, with Corning Gorilla Glass 3
- **Connectivity:** Micro USB, SlimPort, NFC, Bluetooth 4.0, 2x2 MIMO Wi-Fi 802.11 a/b/g/n/ac (single stream)



Figure 2.4. LG Nexus 5

■ 2.8.2 OBD II devices

One of the features of mobile application is the ability to gather driving data from ECU of the car. For this purpose the application uses standardized solution with OBD dongle equipped with ELM327 chip which is connected to car's diagnostic socket.

■ 2.8.2.1 MINI ELM327 bluetooth OBD2 V1.5 Bluetooth dongle

This dongle (in figure 2.5) is widely available and very cheap (\$10 in December 2015) device that can be used with the mobile application. It was used while testing the functionality in real vehicle (Ford Focus, manufactured in 2008). It can be purchased in global online stores such as Amazon¹⁾ or Ebay²⁾ as well as in local online stores. Beside the reading features it allows to diagnose trouble codes and display their meaning. It can clear these codes and turn of the 'Check Engine' light. These features are not implemented in current version of the mobile application. However, this device suffers from a big problem - it does not disconnect when engine is not running and therefore it may drain the vehicle's battery.

¹⁾ <https://amazon.com>

²⁾ <https://ebay.com>



Figure 2.5. MINI ELM327 bluetooth OBD2 V1.5 Bluetooth dongle

Chapter 3

Design

This chapter is dedicated to the design process. First section is about the requirements of the application followed by a proposal of the architecture. In the second section, the user interface design process is described. Last section contains information about a logging server and other tools.

3.1 Requirements

This section describes the essential and most critical features of the designed mobile application. They result from the previous chapter 2 and set a baseline that will be reached in the final mobile application.

- Simplicity of usage - the user flow of the mobile application must be obvious for the user. There is no time in a car for exploring hidden and unclear functionality,
- Low cognitive load - using the mobile application is a secondary task. Therefore, the cognitive load demand (described in section 1.3.3) must be as low as possible,
- Extensibility of functions - the mobile application is designed as a general UI concept with possibility to extend its functionality (e.g., Internet of Things, new in-car information, V2V and V2X communication, etc.),
- Logging feature - as a car is great source of driving data, logging possibilities of the data are highly beneficial.

3.2 Android Design Guidelines

In section 2.6.1 was mentioned that with the release of Android 5 Lollipop, Google provided guidelines for application developers to design user interface in proper way. These guidelines are called Google Design Guidelines ¹⁾ and they defines various design rules and parameter constrains. The most important assumption is that the display of the device should be taken as a real-world 3D environment. The whole concept is called Material Design ²⁾.

3.3 Infotainment user interface architecture proposal

Based on the previous research (led by Dr. Jan Šedivý supervisor of this thesis) the simple user interface structure was proposed. The concept of on-board computers, where the amount of information displayed at a time is limited, was adopted. The general overview of this concept is at figure 3.1.

¹⁾ <https://www.google.com/design/spec/material-design/introduction.html>

²⁾ <https://www.google.com/design/spec/what-is-material/environment.html>

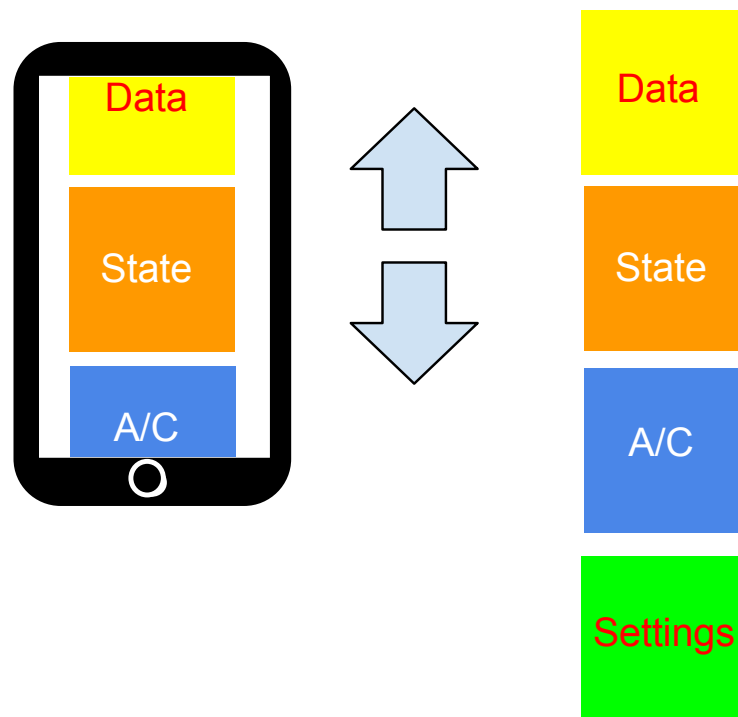


Figure 3.1. Limited information shown at display (credits: Dr. Jan Šedivý)

■ 3.3.1 Portrait vs. Landscape mode

The smartphone changes the user interface based on how the user holds it. The screen in the car can also be in the landscape or portrait mode. The landscape positioning is used by most of the cars on the market. The homepage is showing the icons of the key applications and the user is choosing with the pointing device (touchscreen). The icons are fixed or move towards the central screen position. The horizontal pointing is used to move to both sides. The landscape position is on the other side not good for displaying list of names. It is easy to prove this statement: When a user searches a person in a contact application we can see many more lines in the portrait mode.

The homepage can be designed for both directions the same way, simply moving the focus left to right or up and down. For the radio application, where probably the number one action is changing the station, probably the better format is portrait, since larger number of stations can be displayed. The media application is not that clear: the forward, fast forward or backward, and stop buttons are traditionally arranged horizontally. On the other hand the selection of an artist, song, album, genre etc. will be better in the portrait mode. The navigation map uses in most cases the landscape screen, but again to select the destination is a selection from a list where the portrait positioning might give an advantage. As we have mentioned above, for the phone is probably better the portrait mode. The last key application is the setting of the car parameters. In this case we deal with large number of parameters, where a portrait mode may have an advantage.

It is not easy to lean on either side in selecting the screen positioning. However strictly sticking to one direction of moving between elements on the screen, going between icons, and selecting from a list of text may have an advantage. If the horizontal or vertical approach would be chosen, the UI might be more intuitive and easier to learn. A

portrait arrangement with possibility to move around icons and change lines in lists of names only up and down can be easily controlled by two buttons: up and down.

The portrait mode may be also easier to move in between the speedometer, tachometer, and current RPM value. Even if the portrait mode is rare it may have some advantages in the amount of information on the screen. However the final selection of the screen positioning is a usability study problem.

■ 3.3.2 Use cases

This section is dedicated to observation of use cases in which the application will be used. The primary goal of the application is to be used while driving. However, some initial configuration must be done as well. As it would be very hard, time demanding, and even dangerous to set the configurations while driving, better approach is to introduce the second goal. There are two most important use cases for which will be this application designed.

■ 3.3.2.1 Using while driving

This use case is supposed to radically predominate. A smartphone is placed in a holder, firmly attached to a car's dashboard. Vehicle is moving and user is interacting with the smartphone using one hand and simple gestures while he is still managing the primary task – driving.

■ 3.3.2.2 Setting up

Second use case aims to the setup part. This use case is applied when the vehicle is not moving and the user does not have to pay attention to external environment. In this use case, initial setup (Bluetooth connection to a OBD II Dongle similar to the described one in section 2.8.2) is done.

■ 3.4 Graphical User Interface

User interface is one of the most important part of this thesis. User interface is one of the main tools of communication between an application and the user. User is the key factor in designing a user interface - if a user interface design is bad, nobody will use the application.

There are four basic principles of user interface design known since 1971 [29]:

- Know the user,
- minimize memorization,
- optimize operations,
- engineer for errors.

For the design of a user interface, there have been adopted well-known software engineering processes. For the purpose of this thesis, the simplified iterative model was used. The design of user interface went from the simple sketches to slightly more complex low fidelity prototype to high fidelity prototype which led to the final application. Every phase was evaluated and possibly improved. Throughout the whole process, the notes, and advices from [30] were used.

■ 3.4.1 Low fidelity prototype

Low fidelity prototypes are generally prototypes of limited functionality and interaction. They are constructed to depict concepts, design alternatives, and screen layouts.

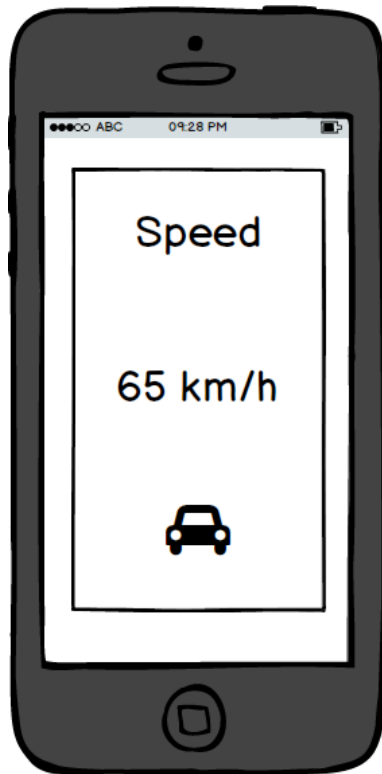


Figure 3.2. Low Fidelity mock up version 1

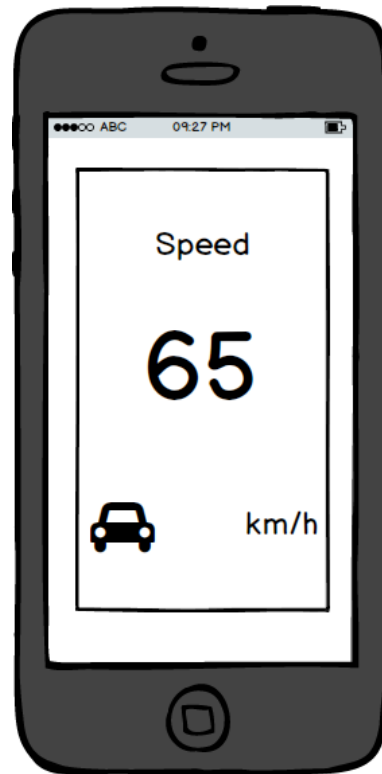


Figure 3.3. Low Fidelity mock up version 2

They are constructed quickly and provide limited or no functionality. These prototypes are created to demonstrate the general look, they are not intended to show in detail how the application operates. They should communicate, educate, and inform, not train, test, or serve as a basis from which to code [31].

For the purpose of creating low fidelity prototype of the application, the tool Balsamiq Mock-ups¹⁾ was used. It is a rapid wireframing tool that reproduces the experience of sketching on a whiteboard, but using a computer.

The two step approach was applied. In the first step the ideas from previous chapters were used and the first version of lo-fi prototype was sketched. The result is in figure 3.2. After discussion and basic testing the mock-up was improved and the second version was designed (figure 3.3).

■ 3.4.2 High fidelity prototype

Unlike low fidelity prototypes, high fidelity prototypes are fully functional and interactive. They address the issues of navigation and flow and of matching the design and user models of a system. High fidelity prototypes are used for exploration and tests. Users can operate them the same way as the final product and they can make informed recommendations about how to improve the user interface [31].

High fidelity prototype of the application developed in this thesis was made in the target application's platform IDE and continuously improved.

¹⁾ <https://balsamiq.com/>

3.5 Logging server architecture

Server is an additional part to the mobile application. It has only one purpose – to store log files generated by the mobile application. However, the solution is required to be robust and scalable. The server solution is proposed to be a simple Java HTTP Servlet for managing the communication with mobile clients and Mongo DB database to store the clients' log files.

3.5.1 Java Servlet API

A Java servlet is a Java class that extends the capabilities of an application server. Servlets can respond to different types of requests. The most common use of servlets is with the HTTP protocol¹⁾. Typical HTTP servlet lifecycle is in figure 3.4²⁾.

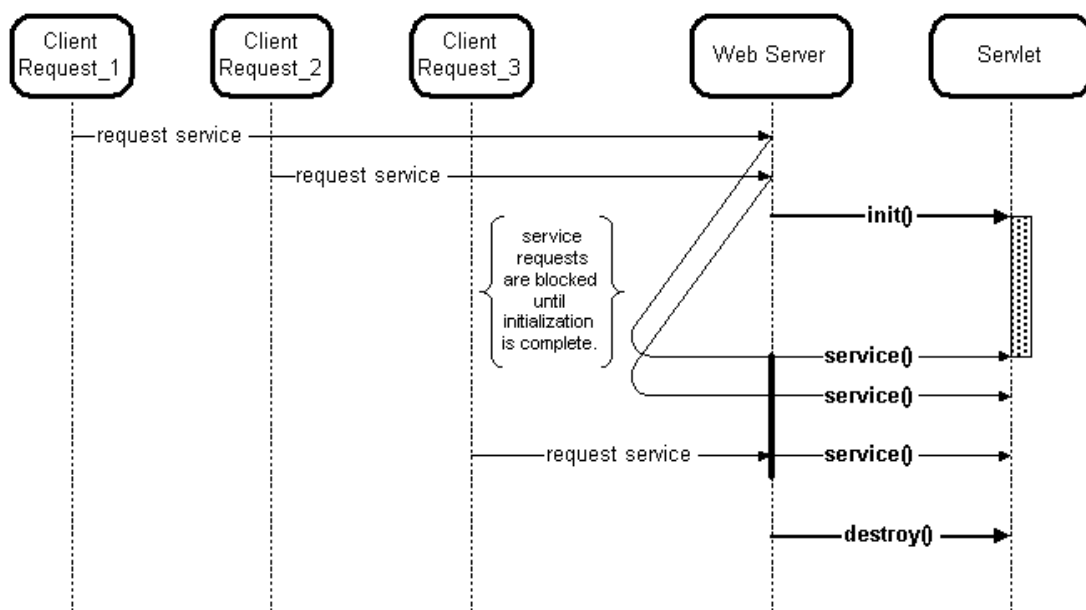


Figure 3.4. Servlet life – flow diagram

3.5.2 Mongo DB

MongoDB is a cross-platform document-oriented database. It is classified as a NoSQL database. This properties makes it a perfect candidate for storing data-intensive log files generated by cars.

¹⁾ <https://www.jcp.org/en/jsr/detail?id=315>

²⁾ https://www.cse.iitb.ac.in/infolab/Data/Courses/DBIS/Software/servlets/servlet_life.gif

Chapter 4

Implementation

In this chapter the main information about development of the mobile application CarDashboard and logging server will be introduced. Firstly, used developer tools will be presented. The mobile application and logging server structures will follow. At the end of this chapter, an integration of mobile application and the logging server will be described.

4.1 Development environments

4.1.1 Android SDK

The Android SDK is a set of various tools that help developers to develop mobile applications for the Android platform. The tools are divided into two groups: SDK tools and platform tools. SDK tools are independent on target Android platform, platform tools are customized to support concrete version of Android OS [32].

The SDK tools consist of Virtual device tools (such as Android Virtual Device Manager or Android Emulator), Development tools (such as SQLite DB Explorer, SDK Manager or Translation editors), Debugging tools (Android Monitor, Android Debug Bridge, etc.), Build and Image tools.

4.1.2 Java Platform, Standard Edition

Java SE is widely used platform for development and deployment of portable code for desktop and server environments. It uses the object-oriented programming language called Java. Java SE defines general purpose APIs, such as Java APIs for Java Class Library. It also includes the Java Language Specification and Java Virtual Machine Specification [33]. There are many implementation of Java SE, most known are Oracle's Java Development Kit (JDK) and OpenJDK.

4.1.2.1 Oracle v. Google

Android architecture has been using a Java Class Library derived from the Apache Harmony¹⁾ project supported by IBM. After Sun Microsystems was bought by Oracle in 2011, support of Apache Harmony ended (IBM joined OpenJDK) and Oracle sued Google for breaching the copyright law (by using Java APIs in Android)²⁾.

With the new version of Android OS – Android L – Google is switching to OpenJDK to bring support of new features implemented in last versions of Java (e.g., Lambda operators)³⁾.

¹⁾ <http://harmony.apache.org/>

²⁾ <https://www.eff.org/cases/oracle-v-google>

³⁾ http://www.theregister.co.uk/2015/12/30/android_openjdk/

■ 4.1.3 Java Platform, Enterprise Edition

Java Platform, Enterprise edition (Java EE) is the standardized platform in community-driven enterprise software which extends Java Standard Edition. Java EE is developed using the Java Community Process¹⁾ with contributions from industry experts, commercial, and open source organizations and many individual developers. Nowadays there are more than 20 compliant Java EE implementation [34].

■ 4.2 Development tools

Development tools allow developers to implement, test, and debug their code. The whole development process is possible even without these tools, but it would take more time to gain the same results. During the development of the mobile application and other software tools, different development tools were used.

■ 4.2.1 Integrated development environment

For the purpose of this thesis the official IDE for Android application development provided by Google was used. Android studio is based on IntelliJ IDEA, a commercial product by the company JetBrains s.r.o. Beside the standard set of functions from IntelliJ IDEA, Android Studio offers tailored-made extensions for Android application developers, therefore it is the recommended option to start the development with [35].

■ 4.2.2 Code version control system

Version control is a system that records changes to a file or set of files over time so that user can recall specific version later. Version control is mostly used in software development process to manage different versions of source code. But VCS is not limited only to plain text information. Graphics and other multimedia can be versioned as well [36].

There are three main approaches in version controlling:

- **Local Version Control Systems** - all changes are stored only on the single computer used by user (e.g., simple file copying to other folder),
- **Centralized Version Control Systems** - versions are stored on the server and all clients access it from remote locations (e.g., Subversion, CVS),
- **Distributed Version Control Systems** - versions are stored on the server, but all clients have local copy of the repository and are able to make changes independently (e.g., Git or Mercurial).

■ 4.2.2.1 Git

For the purpose of development the application, Git VCS was chosen. Git was founded in 2005 by the Linux development community (in particular Linus Torvalds, the creator of Linux) to be an alternative to the commercial product BitKeeper which was used as VCS for development of the Linux kernel. The main objectives were [37]:

- Speed,
- Simple design,
- Strong support for non-linear development (thousands of parallel branches),
- Fully distributed,
- Ability to handle large projects like the Linux kernel.

¹⁾ <http://www.jcp.org/>

■ 4.2.2.2 GitHub

GitHub¹⁾ is a web-based Git repository hosting service. It was used for all the developed software applications and tools in this thesis:

- Android application for smartphone²⁾
- Shared core for Android platform³⁾
- Logging server⁴⁾
- OBD Bluetooth Emulation tool⁵⁾
- Thesis T_EX source⁶⁾

■ 4.2.3 Continuous Integration Service

Continuous Integration (CI) is a software development practice where developers are integrating their work frequently. It is used in development teams of all sizes. Usually the code is automatically verified, tested, and built to prove that changes do not have any hidden side effects. It is very important to set up whole build process properly. CI is often connected to VCS. For the purpose of development the mobile application, Travis CI⁷⁾ was used.

■ 4.2.4 Build Automation System

As mentioned in previous section, automated builds are important in CI process. But well-prepared Build automation system can save a considerable amount of time. Automated builds are compiling source code, resolving dependencies, packaging binary code, and running automated tests. All these task are executed directly from developer's computer or remotely. Most known Build automation systems are: GNU Make⁸⁾, Apache Ant⁹⁾, Apache Maven¹⁰⁾, and Graddle¹¹⁾.

■ 4.3 Third party libraries

During the development of the application and logging server the third party libraries were used. The following sections contain names of the libraries (or dependencies) and reason why they were used in the project.

■ 4.3.1 Mobile application

In mobile application third party libraries were used mainly for the purposes of communication (with a car ECU or with server). One library also implements drag and drop feature, which is described in 4.4.1.

- OBD-II Java API¹²⁾ - support for OBDII PIDs in serial communication,

¹⁾ <https://github.com/>

²⁾ <https://github.com/eclubprague/CarDashboardPhone>

³⁾ <https://github.com/eclubprague/CarDashboardCore>

⁴⁾ <https://github.com/lukashruby/cardashboard-logging-server>

⁵⁾ <https://github.com/lukashruby/obd-mocked-bluetooth-dongle>

⁶⁾ <https://github.com/lukashruby/diploma-thesis>

⁷⁾ <https://travis-ci.org/>

⁸⁾ <https://www.gnu.org/software/make/>

⁹⁾ <http://ant.apache.org/>

¹⁰⁾ <https://maven.apache.org/>

¹¹⁾ <http://gradle.org/>

¹²⁾ <https://github.com/pires/obd-java-api>

- OkHttp¹⁾ - an HTTP client for Android applications, used for communication with server,
- DragSortListView²⁾ - implementation of drag and drop used for module management.

■ 4.3.2 Logging service

As logging server receives files from mobile clients and saves the content to the database, third party libraries were used to simplify development of these features.

- Commons FileUpload³⁾ - robust, high-performance, file upload capability for servlets and web applications,
- MongoDB Java Driver⁴⁾ - provides synchronous and asynchronous interaction with Mongo databases.

■ 4.4 Mobile Application Implementation

Mobile application is divided into two parts. The first part is platform dependent (smartphone in this case) and will be described in detail in section 4.4.1. The second part, so-called **Core**, is platform independent, and can be ported to other devices which use Android OS (tablet, smart tv, etc.) In [38] this part is used in tablet version of this application. **Core** was developed in cooperation with the author of [38]. Details of the contribution is available in Core's Github repository⁵⁾. From this part of project, only the important parts will be described in 4.4.2.

■ 4.4.1 User interface for smartphones

Design of the user interface was the most important part of chapter 3 and it is a key problem of the whole project. As this application is aimed to be used in a car while driving, bad implementation of the concept can cause many difficulties for drivers. Therefore it was very important to be especially precious in the development and this phase took the majority of time.

Standard mode of application consists of various **Modules**. Mobile phone version displays only one module at a time (tablet version [38] contains more modules at the screen). These modules are ordered in a tree structure – a single module is displaying some information or contains more modules.

Beside of the standard mode of application, when phone is placed firmly in holder and a driver uses single finger to change views, there is also the configuration mode. In the configuration mode there are options to enable or disable *Bluetooth support* (for connection to OBD II Dongle) and to configure *Bluetooth devices*. Also the settings of *Logging features* is present in this mode. Last but not least function is *Module management*. Each part of the application will be described in dedicated subsections.

■ 4.4.1.1 ScreenSlideActivity & ScreenSlidePageFragment

Prevailing part of user interface is the implementation of vertical scrolling list of modules in **ScreenSlideActivity**. This list is presented as infinite menu (after last item in list comes the first one in loop). Android provides support only for horizontal scrolling list,

¹⁾ <https://github.com/square/okhttp>

²⁾ <https://github.com/bauerca/drag-sort-listview>

³⁾ <https://commons.apache.org/proper/commons-fileupload/>

⁴⁾ <https://mongodb.github.io/mongo-java-driver/>

⁵⁾ <https://github.com/eclubprague/CarDashboardCore>

so this implementation was done from scratch. `ScreenSlidePageFragment` is then the content of single Module. It consists of a Module name, a Module icon, a Module value, and a Module units.

■ 4.4.1.2 DnDActivity & DnDFragment

DnD means in this case abbreviation for Drag and Drop. This feature is used in configuration part, where user can change order of modules, add new modules, and delete modules with simple swiping gestures. This approach makes the configuration process easy and straightforward.

■ 4.4.2 Shared Core Library

The core contains all the functionality, it handles data, logic and also a standardized part of presentation, which consists of predefined single module views. It includes also the communication settings with the car's ECU and a service for uploading log files to the server. This library is described in detail in [38],

■ 4.4.3 Final application

The final application screenshots are presented in the following set of figures: Figure 4.1 shows the typical single information display module (at this case with the current speed of car), figure 4.2 is a demonstration of a folder module. This module contains another set of modules (just like folder structure of computer's filesystem). Figures 4.3, 4.4, 4.5, and 4.6 displays the interaction with the application.

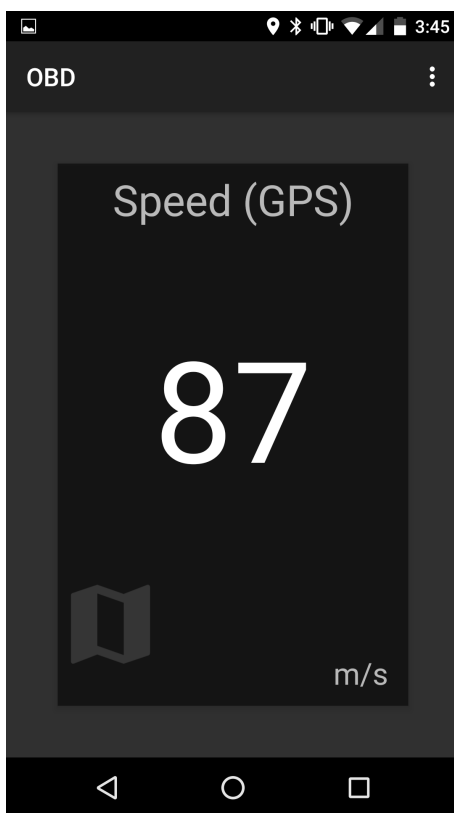


Figure 4.1. Module displaying single information.

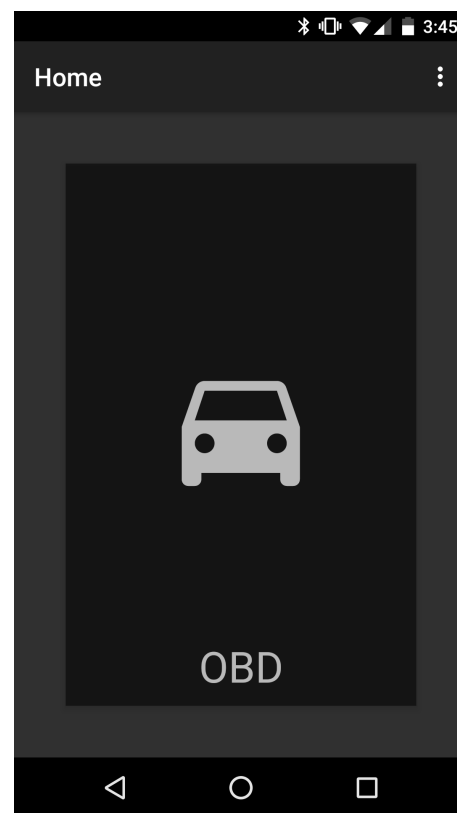


Figure 4.2. Module contains another set of child modules - folder.

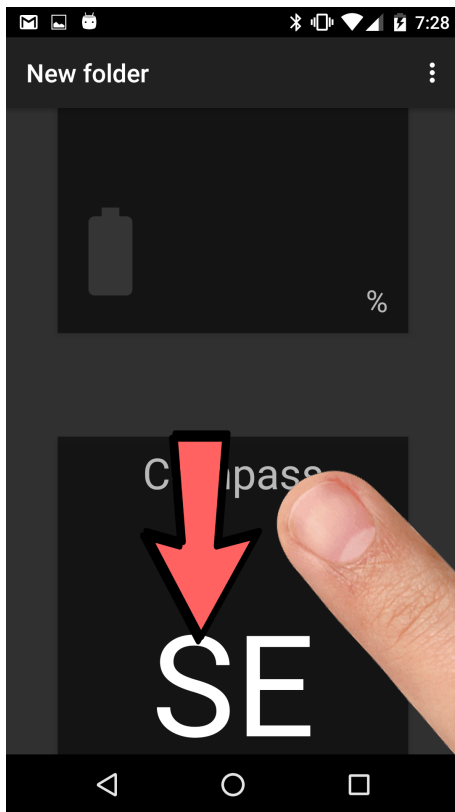


Figure 4.3. Swiping between different modules.

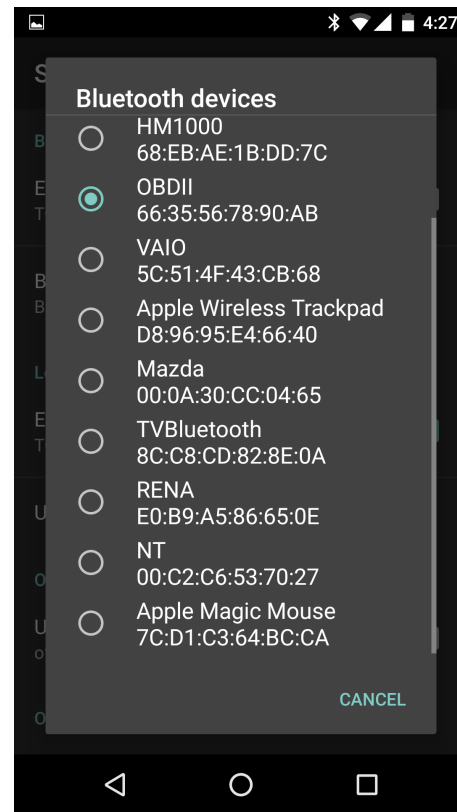


Figure 4.4. Selection of Bluetooth device to connect to OBD dongle.

4.5 Server implementation

An important part of the assignment of this thesis is implementation of a logging server. Purpose of the server is to gather data from devices with installed mobile application and provide these data to further exploration. Server was implemented as HTTP service in Java EE using Servlet API. It can be deployed to various application servers such as Apache Tomcat or Oracle Glassfish. Oracle Glassfish¹⁾ was used for deployment in this thesis.

4.5.1 HTTP servlet

HttpServlet is an abstract class to be subclassed to create an HTTP servlet suitable for a HTTP protocol. A subclass of HttpServlet must override at least one method, usually one of these:

- doGet, if the servlet supports HTTP GET requests
- doPost, for HTTP POST requests
- doPut, for HTTP PUT requests
- doDelete, for HTTP DELETE requests
- init and destroy, to manage resources that are held for the life of the servlet
- getServletInfo, which the servlet uses to provide information about itself

¹⁾ <http://www.oracle.com/us/products/middleware/cloud-app-foundation/glassfish-server/overview/index.html>

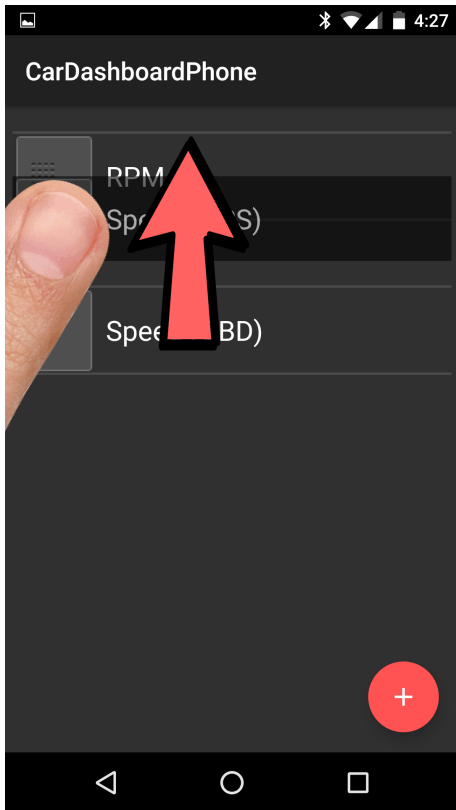


Figure 4.5. Changing position of module using Drag and drop.

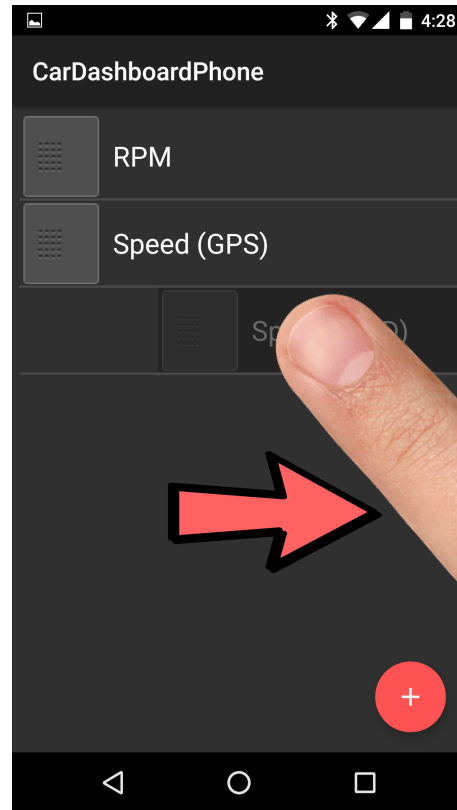


Figure 4.6. Removing module using simple swipe gesture.

■ 4.5.2 Communication with server

Mobile application is logging all OBD data to the file stored in local storage. As these logs may be very data intensive, it is not reasonable to upload them live. Therefore when a wi-fi connection is available, user can initiate the upload of all files to the Java-based server running in a cloud. These communication is implemented using the standard multipart POST request.

Chapter 5

Evaluation

Mobile application developed in this thesis is supposed to be used while driving and drivers must interact with it to obtain the desired information. This interaction is critical, as it might cause serious problems while driving. The aim of this chapter is to show that the mobile application can be used without significant impact to driver's behaviour.

The primary goal of the evaluation part was to collect data from real human drivers in the tasks and environment as close as possible to real-world situations. To reach this goal we established cooperation with the Driving Simulation Research Group based at the Faculty of Transportation Sciences of Czech Technical University in Prague. We were able to use their car simulator and other equipment.

5.1 Usability tests

Usability testing is a good way to understand how real users use an application. Unlike the other methods (interviews, focus groups, etc.) usability testing measures actual performance on given tasks. In case of infotainment mobile application it is crucial for users to drive safely. All usability tests were primarily designed to evaluate the cognitive load of driver (as the application was designed to minimize it).

5.1.1 Used equipment

As mention at the beginning of this chapter, we used equipment provided by the Faculty of Transportation Sciences, namely the 3D car light simulator and Eye Tracker. Logged results were analyzed using MATLAB.

5.1.1.1 3D light simulator Skoda Octavia II

Simulator was build by DSRG. It is based on cockpit of Skoda Octavia II with cave-like projection system. Inner cockpit parts are cut jut behind the drivers seat. The cockpit is fully equipped like the normal car (including roof, co-driver seat, etc.). Picture of the simulator is at figure 5.1. Transmission was 6-speed automatic gearbox.

5.1.1.2 Software tools

The developed mobile application requires the Bluetooth connection to obtain driving data from OBD dongle. As the simulator was not equipped with standard CAN-BUS communications (and therefore neither the OBD protocol was supported), using the common way of accessing driving data was not possible. To by-pass this problem, a support software tool in Java was developed. This program reads a subset of driving data (provided by the simulation software) from network and simulates an OBD dongle. The mobile application is then connected to the target computer and is able to read data the same way as if it was connected to the OBD dongle.



Figure 5.1. Picture of simulator

■ 5.1.1.3 Eye Tracking Cameras & Software

Eye tracking solution was used for measuring the time that was spent looking at the mobile phone while driving. Car Simulator were equipped with the product of the company Smart Eye AB¹). It was composed of two eye tracking cameras Smart Eye Pro²). These cameras are designed to be used in Car Simulators, Flight Deck Simulators, and other vehicle studies. The cameras are operating under Infra-Red, so they are insensitive to ambient light.

Another part of the eye tracking solution was Smart Eye Pro software. This program was used to configure the whole process. Every participant went through the calibration procedure. Screenshot from the program is in figure 5.2.

■ 5.1.2 Participants

The usability tests are based on the results from 5 participants from age 24 to 45 (mean age was 28, standard deviation 9.62). One participant was female, other four were males. Kilometers drove per year by the participants were in range from 4000 to 30000 (mean of 14800, standard deviation 10849). Operating system used on smartphone was three times Android OS, one time iOS, and one time Windows Phone. Other information retained from participants are in table 5.1.

The number of participant was based on recommendation of Jakob Nielsen [39]. None of participant had previous experience with the tested applications.

■ 5.1.2.1 Screener

Purpose of the screener questionnaire was to filter out the participants that did not meet the requirements. The requirements were simple - a participant must be a smartphone user and active driver. All questions with required answers are available in section C.1 of appendix C.

¹) <http://smarteye.se/>

²) <http://smarteye.se/products/smart-eye-pro/>

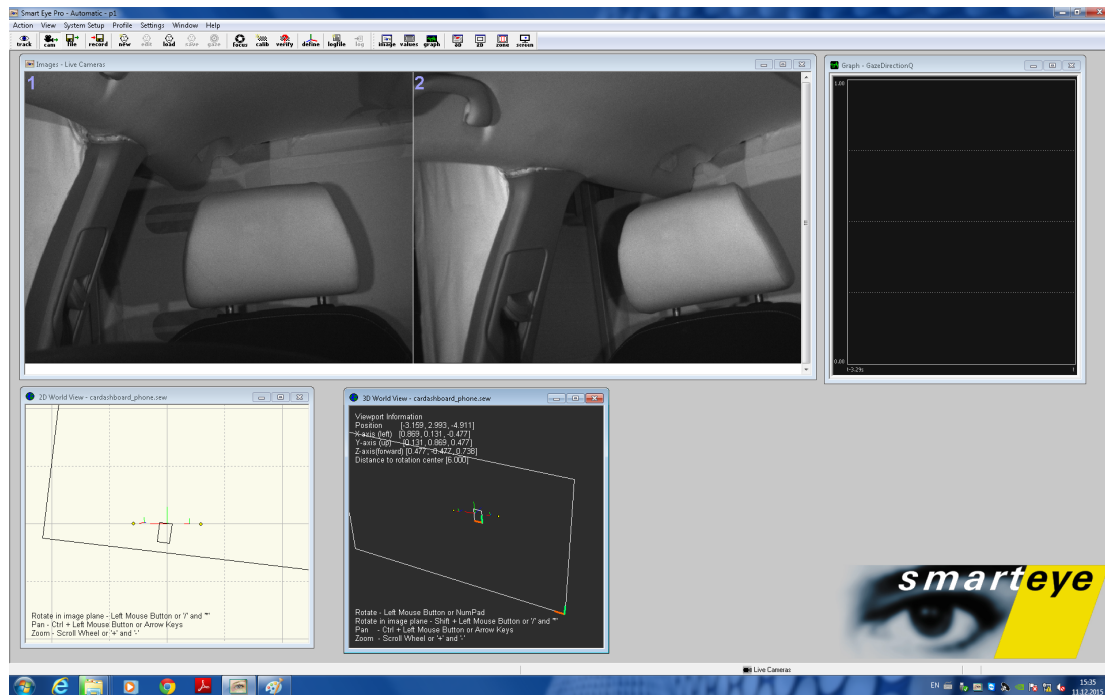


Figure 5.2. Eye tracking software

Participant's #	Sex	Age	Times per week driving	Kms per year	Smartphone OS	Use phone while driving?	How often?
Part. 1	Male	24	1	4000	WP	Yes	Occasionally
Part. 2	Male	26	5	30000	Android	Yes	Often
Part. 3	Female	45	7	20000	iOS	Yes	Often
Part. 4	Male	23	2	5000	Android	Yes	Occasionally
Part. 5	Male	22	6	15000	Android	Yes	Often

Table 5.1. Results of pre-test questionnaire

5.1.2.2 Pre-test questionnaire results

Before execution of the usability test, participants were asked to answer some questions about their behavior while driving. The questionnaire is available in table C.2, results are in the table 5.1.

5.1.2.3 Post-test questionnaire results

After the end of every single test, participants answered few questions about their impression about the process, their understanding, and generally about their feeling about using both applications. Questionnaire is available in section C.3 of appendix C, results are summarized in table 5.2.

Participant's #	Impression of using the device while driving [1-5]	Goals were clear [yes/no]	Way of solution was acceptable [1-5]	Amount of displayed information was appropriate [1-5]	Would use the application in everyday driving [yes/no]
Part. 1	1	yes	1.5	1	yes
Part. 2	3	yes	3	1	no
Part. 3	2	yes	1	1	yes
Part. 4	1	yes	2	1	yes
Part. 5	2	yes	1.5	2	yes

Table 5.2. Results of post-test questionnaire

5.1.3 Tests schedule

Each test had a fixed time limit. As was mentioned in the previous section, participants had not had any experiences with in-car dashboard mobile application and they had not been participants of usability testing before. Therefore an introduction part was included where they were informed about structure and purpose of the test. This introduction was followed by necessary calibration of eye-tracking cameras (for each participant the calibration was different). Then the usability tests could be performed. Exact plan of a test is shown in table 5.3.

Start	End	Duration [min]	Content
00:00:00	00:05:00	5	Introduction
00:05:00	00:10:00	5	Pre-test questionnaire
00:10:00	00:25:00	15	Instructions and EyeTracker setup
00:25:00	00:40:00	15	Warm-up driving
00:40:00	00:55:00	15	A/B testing
00:55:00	01:05:00	10	CLT testing
01:05:00	01:10:00	5	Post-test questionnaire
01:10:00	01:15:00	5	Debriefing

Table 5.3. User testing schedule

5.2 Lane Change Test

This test was used to evaluate influence of a secondary task while performing a primary task. For these purposes the Lane Change Test (LCT) has received considerable attention. It appears to be a practical and effective measure of the costs associated with operating in-vehicle devices while performing a driving task. The procedure of the Lane Change Test was standardized as ISO 3888.

For the execution of the LCT the car simulator introduced in section 5.1.1 was used. The test track was a 4 km straight line road with a single lane change maneuver. Participants were required to keep a constant speed of 60 km per hour and they were asked to tell the current value of a set of driving-relevant information displayed on the smartphone using the developed application every 20 seconds. The interval between each request was 10 seconds. The maneuver was initiated by placing an unexpected object in the driving path. Size of the object was 2.20 x 2.20 x 2.20 m and it was placed 35 m in front of vehicle. There was a red arrow pointing to the left lane which suggested driver the direction to go around the object (change the driving lane).

5.2.1 Results

Evaluation of Lane Change Test is based on two metrics. The first one is the object avoidance (if a participant did not hit the object) and the second one is the reaction time. This value is determined from the usage of steering wheel - when steering angle was bigger than certain value (calculated from previous drive). All data necessary for evaluating the tests were obtained from car simulator. Preview of typical behavior is plotted in figure 5.3.

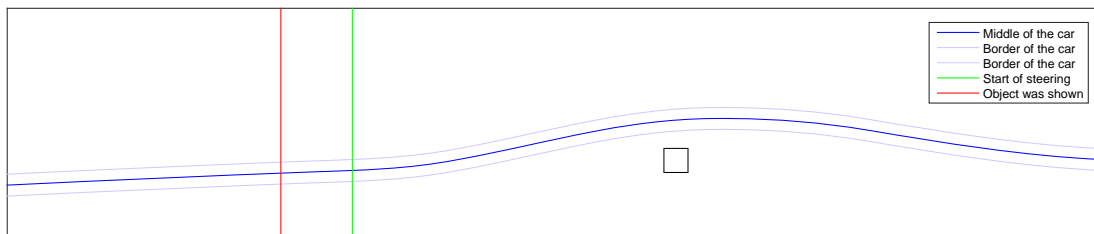


Figure 5.3. Typical process of object avoidance during LCT

Participant #	Object avoided	Reaction time	Reaction distance
Participant 1	Yes	437.1 ms	8.91 m
Participant 2	Yes	491.0 ms	9.39 m
Participant 3	Yes	455.5 ms	9.10 m
Participant 4	Yes	587.5 ms	10.99 m
Participant 5	Yes	511.8 ms	9.87 m

Table 5.4. Results of LCT

Table 5.4 shows that in the object avoidance the tests were 100% successful. However, the result with bigger group of participant could be different - for example in picture D.4 in Appendix D is obvious that object avoidance was very tight.

Measured and calculated results can be compared with results from [40]. A similar test was performed with measured mean time 0.78 seconds. This reaction time is comparable with reaction times of participants.

5.3 A/B test

A/B testing (also known as split testing or bucket testing) is a testing approach of comparing two version of tested subjects again each others to determine which one performs better. This approach is known mainly from webpage testing, where the criterion is conversion rate. Method of A/B testing was adopted also in testing of CarDashboard mobile application. Small changes were done: Instead of webpages were compared mobile applications and instead of conversion rate the criterion was glance time. Glance time was measured using eye tracking cameras mentioned in section 5.1.1. Same as in the previous test, participants were asked to tell the current values of a set of driving-relevant information displayed on the smartphone using the developed application with the frequency of 10 seconds. The requested information types changed after each iteration, so the participant must interact with the device after each request.

A/B testing needs two subject. In this case, the first subject was the developed mobile application CarDashboard and the second subject was a mobile application called Torque¹⁾ which is the most downloaded application for OBD connection with more than 1 million free downloads²⁾ and more than 500 thousands paid downloads³⁾. This application is available for Android-based tablets and mobile phones, thus was a perfect candidate as the usability tests were performed also for the tablet version of CarDashboard [38].

For purposes of A/B testing, a complex world map was used. The whole route was 6.5 km long and led through countryside with two villages. Also the other traffic was present. The 2D map of the route is displayed in figure 5.4 and typical speed and usage of brakes is in figure 5.5.

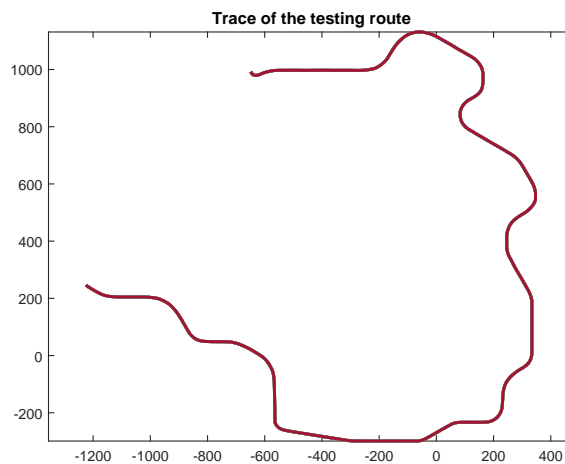


Figure 5.4. Route of A/B tests

5.3.1 Results

The hypothesis of the A/B testing was that the developed mobile application CarDashboard would perform better than the competitive application Torque. This hypothesis was confirmed. All participants also proclaimed that the CarDashboard application is simpler to use and they had better feeling of use. All results from A/B testing are

¹⁾ <http://torque-bhp.com/>

²⁾ <https://play.google.com/store/apps/details?id=org.prowl.torquefree&hl=en>

³⁾ <https://play.google.com/store/apps/details?id=org.prowl.torque&hl=en>

available in table 5.5. The results are considered as expectable. For some participants the difference between Torque and CarDashboard is not significantly diverse, however, the rating score of Torque on Google Play Store must be taken in account. Torque reached the score 4.0 from 5.0 (from over 21 thousands reviewers) and therefore it is a very good rated application. Torque Pro has even better score – 4.6 from almost 40 thousands reviewers.

Participant's #	Application A Glance Ratio	Application B Glance Ratio	Application A Maximal Glance	Application B Maximal Glance	Application A Average Glance	Application B Average Glance
Participant 1	18.63 %	11.2 %	1283 ms	1150 ms	182.47 ms	138.89 ms
Participant 2	15.72 %	10.72 %	1602 ms	1129 ms	172.76 ms	140.42 ms
Participant 3	8.90 %	7.78 %	450 ms	533 ms	70.2 ms	66.67 ms
Participant 4	13.94 %	5.4 %	1933 ms	433 ms	541.53 ms	187 ms
Participant 5	9.94 %	10.32 %	950 ms	1617 ms	73.41 ms	120.7 ms

Table 5.5. Results of A/B testing

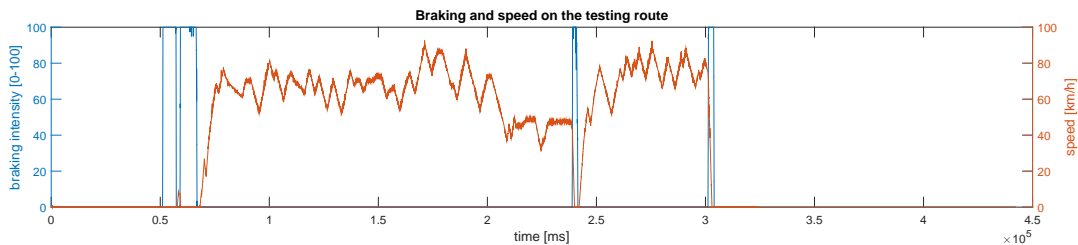


Figure 5.5. Visualization of typical speed and braking during A/B test

Chapter 6

Conclusion

In the last chapter of this thesis, the assignment competition will be analyzed. For every task, the relevant section will be referenced. After this analysis will come the section that summarizes the general work on this thesis and its consequences. The last section of this chapter (and thesis) is dedicated to future plans.

6.1 Assignment completion

6.1.1 Completing the assignment tasks

6.1.1.1 Study the current trends in controlling car infotainment systems

Current trends in design and control of infotainment systems are summarized in section 1.3.1. Historical evolution and current state are compared. The complexity of currently used devices in in section 1.4.

6.1.1.2 Explore existing mobile applications for in-car use

The most significant producers of mobile applications for in-car use are mentioned in section 2.3. In the chapter 2 is overall described current market, the shipments of mobile devices, and a variety of available mobile operating systems.

6.1.1.3 Study communication with car ECU (CAN, OBD, etc.)

Data and communication inside a vehicle is described in section 2.4. Different types of in-car buses are listed in section 2.4.1. Moreover in section 2.4.3 is also the analysis of communication with the outer world.

6.1.1.4 Design and implement mobile application for in-car user

This task is the topic of chapter 3 and 4. In chapter 3 the design process is thoroughly described including the general guidelines. In chapter 4 the implementation phase is discussed. It contains description of used environment, software tools, third party libraries, and the mobile application implementation.

6.1.1.5 Implement logging server

Process of design and development of the logging server for purposes of storing driving data is described in section 2.7, where the different concepts of cloud computing are introduced, then in section 3.5 the logging server architecture is proposed and finally in section 4.5 the implementation is described.

6.1.1.6 Integrate mobile application with logging server

Basics of possible ways of communication are listed in section 2.4.3 and particular method used in the development is described in section 4.5.2.

■ 6.1.1.7 Compare your solution with existing solutions

Comparison with the biggest competitor, application Torque, is presented in section 5.3. This comparison was done using the A/B tests on a car simulator with users.

■ 6.1.1.8 Evaluate the application using usability testing

Two types of usability tests were performed. A/B tests for the purpose of compare the application CarDashboard with the competitor. Second type of tests, Lane Change Test, was executed on a car simulator as well. Progress and results of usability testing are available in section 5.2, 5.3, and in appendix D.

■ 6.2 Summary

The goal of this thesis was to examine a new approach in development of in-car mobile applications. This goal was successfully reached. Along the way of solving the particular tasks, many other tools were developed to support the main goal, such as shared library CarDashboard Core, OBD II emulation Java tool and Logging server. These tools can be used even without the main mobile application CarDashboard.

My first contact with the design an in-car user interface and the measurement of cognitive load was during my Erasmus studies at Johannes Kepler University in Linz, Austria. I took the course Principles of Interaction led by Prof. Dr. Andreas Riener, leading expert in this discipline. At those time I did not fully understand all details of this field. Still, this topic was very interesting and after my return to the Czech Republic I decided to carry on under supervision of Dr. Jan Šedivý.



Figure 6.1. Final CarDashboard mobile application with the day mode color scheme

Although the mobile application CarDashboard performed very well in usability test in a car simulator, in-car environment is not the only way where the application can be utilized. After minor changes, it can be used as an universal control panel for the concept of smart building.

During the work on this thesis and relevant project, I have learned various of new skills. First of all, I have got the perfect overview of the Android platform and how to make applications for it. Secondly, I have explored different ways of integration of software systems in local area and wide area networks. In third place, I have started to follow the current trends in automotive user interfaces and discovered that this field is much bigger then I had expected. And lastly, I have had to defend my idea and solutions, discuss them and improve them on numerous meeting with my supervisor and my colleges. Another interesting experience was setting up a cooperation with people from Faculty of Transport Sciences of CTU in Prague, planning the way of usability test execution and solving other challenging problems connected with usability testing. We are on a good way to continue with this so-far successful cooperation.

In the end, I would like to notice it was a great experience working on this topic and I would like to stay in touch with this field in further phases of my life.



Figure 6.2. Final CarDashboard mobile modified for use in smart buildings

6.3 Future work

Even though a large amount of work was done, there are still some tasks left. The most important task that will come in next weeks after defending this thesis is the publishing of the mobile application CarDashboard. The new name will be invented and a new graphic identity of whole application (logotype, colors, etc.) will be introduced. Based on the results from usability tests, we are expecting the application to be successful.

Application will be published on the official application store by Google – Google Play Store and every owner of an Android based smartphone will be able to download it.

After bringing the application to the general public, many new issues with different device combinations will certainly appear. But that is the destiny of every mobile application. We will do our best to provide our users the best possible experience from using the applications.

References

- [1] "European Commission, Digital Agenda for Europe". *"eCall: Time saved = lives saved"*. "2015".
"<https://ec.europa.eu/digital-agenda/en/ecall-time-saved-lives-saved>". Online; accessed 7-January-2016.
- [2] "T-Systems". *"CONNECTED CAR: TRAFFIC AND DIAGNOSTICS."*. "2015".
"https://www.t-systems.com/umn/t-systems-use-case-big-data-automotive-1146850_1/blobBinary/T-Systems-Use-Case_Big-Data-Connected-Car.pdf?ts_layoutId=989624". "[Online; accessed 7-January-2016]" .
- [3] "IHS Automotive, part of IHS Inc. ". *"Average Age of Light Vehicles in the U.S. Rises Slightly in 2015 to 11.5 years."*
"<http://press.ihs.com/press-release/automotive/average-age-light-vehicles-us-rises-slightly-2015-115-years-ihs-reports>". Online; accessed 7-January-2016.
- [4] Tobias Holstein, Markus Wallmyr, Joachim Wietzke, and Rikard Land. *Current Challenges in Compositing Heterogeneous User Interfaces for Automotive Purposes*. Lecture Notes in Computer Science. 2015.
http://dx.doi.org/10.1007/978-3-319-20916-6_49.
- [5] David Canfield Smith, Charles Irby, Ralph Kimball, Bill Verplank, and Eric Harslem. *Designing the Star User Interface The Star user interface adheres rigorously to a small set of principles designed to make the system seem friendly by simplifying the human-machine interface*. 1982.
<http://www.guidebookgallery.org/articles/designingthestaruserinterface>.
- [6] Sonja Rümelin, and Andreas Butz. *How to Make Large Touch Screens Usable While Driving*. In: *Proceedings of the 5th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*. New York, NY, USA: ACM, 2013. 48–55. ISBN 978-1-4503-2478-6.
<http://doi.acm.org/10.1145/2516540.2516557>.
- [7] John Duncan, Phyllis Williams, Ian Nimmo-Smith, and Ivan Brown. *The control of skilled behavior: Learning, intelligence, and distraction*. 1993.
- [8] Christopher D Wickens, John D Lee, Yili Liu, and Sallie Gordon-Becker. *Introduction to human factors engineering*. Addison-Wesley, 1998.
- [9] A.B. Rodriguez Gonzalez, M.R. Wilby, J.J. Vinagre Diaz, and C. Sanchez Avila. Modeling and Detecting Aggressiveness From Driving Signals. *Intelligent Transportation Systems, IEEE Transactions on*. 2014, 15 (4), 1419-1428. DOI 10.1109/TITS.2013.2297057.
- [10] Ranjit Atwal, Lillian Tay, Roberta Cozza, Tuong Huy Nguyen, Tracy Tsai, Annette Zimmermann, and CK Lu. *Forecast: PCs, Ultramobiles and Mobile Phones, Worldwide, 2012-2019, 2Q15 Update*. 2015.
- [11] Christopher D Wickens. *Engineering psychology and human performance ..* Harper-Collins Publishers, 1992.

- [12] Hiroshi Ishii. The tangible user interface and its evolution. *Communications of the ACM*. 2008, 51 (6), 32–36.
- [13] "Google, Inc.". *Android Auto*. 2015.
<https://www.android.com/auto/>.
- [14] "Apple, Inc.". *Apple CarPlay*. 2015.
<http://www.apple.com/ca/ios/carplay/>.
- [15] Konrad Etschberger. *Controller area network: basics, protocols, chips and applications*. Ixxat press, 2001.
- [16] "National Instruments". *FlexRay Automotive Communication Bus Overview*. 2009.
<http://www.ni.com/white-paper/3352/en/>.
- [17] "United States Environmental Protection Agency". "On-Board Diagnostics (OBD)". "1996".
"http://www3.epa.gov/obd/index.htm". Online; accessed 7-January-2016.
- [18] "Google, Inc.". *Android Homepage*. 2015.
<https://www.android.com/>.
- [19] "Apple Inc.". *iOS OS Homepage*. 2015.
<http://www.apple.com/ios/>.
- [20] "Microsoft". *Windows Phone OS Homepage*. 2015.
<https://www.microsoft.com/en-us/windows/phones>.
- [21] "BlackBerry Limited". *BlackBerry OS Homepage*. 2015.
<http://us.blackberry.com/software/smartphones/blackberry-10-os.html>.
- [22] "mozilla.org". *Firefox OS Homepage*. 2015.
<https://www.mozilla.org/en-US/firefox/os/2.5/>.
- [23] "Jolla". *Sailfish OS Homepage*. 2015.
<https://sailfishos.org/>.
- [24] "Linux Foundation". *Tizen OS Homepage*. 2015.
<https://www.tizen.org/>.
- [25] "Canonical Ltd.". *Ubuntu Phone OS Homepage*. 2015.
<http://www.ubuntu.com/phone>.
- [26] "Android Developers". *What is android*. 2011.
<http://developer.android.com/guide/basics/what-is-android.html>.
- [27] Peter Mell, and Tim Grance. The NIST definition of cloud computing. *National Institute of Standards and Technology*. 2009, 53 (6), 50.
- [28] "Wikipedia". "Nexus 5 — Wikipedia, The Free Encyclopedia". "2016".
"https://en.wikipedia.org/w/index.php?title=Nexus_5&oldid=697674134". "[Online; accessed 7-January-2016]" .
- [29] Wilfred J. Hansen. *User Engineering Principles for Interactive Systems*. In: *Proceedings of the November 16-18, 1971, Fall Joint Computer Conference*. New York, NY, USA: ACM, 1971. 523–532.
<http://doi.acm.org/10.1145/1479064.1479159>.
- [30] Matthias Kranz, and Andreas Riener. *Human-Computer Interaction: Design Phase*. Slides from lecture Principles of Interaction. Johannes Kepler University Linz. SS 2014.
- [31] Jim Rudd, Ken Stern, and Scott Isensee. Low vs. High-fidelity Prototyping Debate. *interactions*. 1996, 3 (1), 76–85. DOI 10.1145/223500.223514.

-
- [32] "Android Developers". *Android SDK help*. 2016.
<http://developer.android.com/tools/help/index.html>.
- [33] "Oracle Corporation". *Java SE Overview*. 2016.
<http://www.oracle.com/technetwork/java/javase/overview/index.html>.
- [34] "Oracle Corporation". *Java EE Overview*. 2016.
<http://www.oracle.com/technetwork/java/javaee/overview/index.html>.
- [35] "Android developers". *Android Studio Release Notes*. 2016.
<http://developer.android.com/tools/versions/studio.html>Git.
- [36] Linus Torvalds, and Junio Hamano. Git: Fast version control system. *URL*
<http://git-scm.com>. 2010,
- [37] "Software Freedom Conservancy". *A Short History of Git*. 2016.
<http://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git>.
- [38] Michael Bláha. *Car infotainment application on a tablet (Master thesis)*. 2016.
- [39] Jakob Nielsen. *How Many Test Users in a Usability Study?* 2012.
<https://www.nngroup.com/articles/how-many-test-users/>.
- [40] Pamela D'Addario. *Perception-response Time to Emergency Roadway Hazards and the Effect of Cognitive Distraction*. Ph.D. Thesis, 2014.

Appendix A

Thesis assignment

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Computer Science and Engineering

DIPLOMA THESIS ASSIGNMENT

Student: **Lukáš Hrubý**

Study programme: Open Informatics
Specialisation: Software Engineering

Title of Diploma Thesis: **Car infotainment application on a smartphone**

Guidelines:

The car infotainment systems are reflecting the current trend to digital lifestyle. People are taking their mobile devices to cars and use them while driving. The current applications are not designed for in-car use, they draw too much of driver's attention and they are negatively impacting the road safety. Design a new Android infotainment application minimising the additional cognitive load. Proceed following the next steps:

Examine and explore / study and analyze

- Current trends in controlling car infotainment systems
- Existing mobile applications for in-car use
- Communication with car ECU (CAN, OBD, etc.)

Design and implement

- Mobile application for in-car use
- Logging server

Integrate

- Mobile application with logging server

Compare

- Your solution with existing solutions

Evaluate

- Usability testing

Bibliography/Sources:

- [1] Android Developers Site - <http://developer.android.com/>
- [2] Robert Bosch GmbH: BOSCH Automotive Handbook, Wiley 2004, ISBN: 9781860584749
- [3] Proceedings of the 7th International Conference on Automotive User Interfaces and Interactive Vehicular Applications

Diploma Thesis Supervisor: Ing. Jan Šedivý, CSc.

Valid until the end of the summer semester of academic year 2016/2017

prof. Ing. Filip Železný, Ph.D.
Head of Department



prof. Ing. Pavel Ripka, CSc.
Dean

Prague, November 25, 2015

Appendix B

List of Abbreviations

CAN	■	Controller Area Network
ECU	■	Engine Control Unit
IDE	■	Integrated Development Environment
NIST	■	National Institute of Standards and Technology
OBD	■	On-board diagnostics
OBDII	■	On-board diagnostics, second version
OEM	■	Original Equipment Manufacturer
OS	■	Operating system
PIM	■	Persona Information Manager
RPM	■	Revolutions per minute
UI	■	User Interface
USB	■	Universal Serial Bus
VCS	■	Version Control System
V2V	■	Vehicle-to-vehicle communication
V2X	■	Vehicle-to-anything communication

Appendix C

Usability test questionnaires

C.1 Screener

- Are you a smartphone user?
- Are you a driver?
- How many times per week are you driving?

C.2 Pre-test questionnaire

- Age,
- Sex [male, female],
- How many kms per year are you driving a car?
- Which operating system is your smartphone running?
- Have you ever used your smartphone while driving? [yes, no]
- How often are you using your smartphone while driving? [never, occasionally, often]

C.3 Post-test questionnaire

In [1-5] questions, 1 means the best rating, 5 means the worst.

- What is your impression of using the device while driving? [1-5]
- Were the goals clear for you? [yes, no]
- How acceptable was the way of solution for given tasks? [1-5]
- Was the amount of displayed information appropriate? [1-5]
- Would you use the application in everyday driving time? [1-5]

Appendix D

Usability test results analysis

D.1 MATLAB scripts

D.1.1 Statistics of A/B testing

```
clear all;

load('p4_ab_m_cd.mat');
tablet = [116;97;98;108;101;116];
phone = [112;104;111;110;101];
objectname = phone;
comparation = ismember(ClosestWorldIntersection_objName(
(1:length(objectname)),:),objectname);

watchingPhone = comparation(2,:);
totalPhone = sum(watchingPhone);

glanceRatio = totalPhone / length(ClosestWorldIntersection_objName);

glanceRatio

totalPhone

length(ClosestWorldIntersection_objName)

inputvector = watchingPhone;

glances = [];
currentGlance = 0;
for n = 2:numel(inputVector)
    if(inputVector(n))
        if(inputVector(n-1))
            currentGlance = currentGlance + 1;
        end
    else
        if(inputVector(n-1))
            glances = [glances, currentGlance];
            currentGlance = 0;
        end
    end
end
end
```

```

if(currentGlance>0)
    glances = [glances, currentGlance];
    currentGlance = 0;
end

numOfGlances = length(glances)
maxGlance = max(glances)*1000/60
averageGlance = sum(glances)/length(glances)*1000/60

```

D.1.2 Visualisation of Lane change test

```

steerDiff = find(diff(steer)<-2);
\%steering more than treshold (id, where was the steering bigger)

yL = get(gca,'YLim'); \% length of Y

\%objectX = -5400.7-1.1; objectY = 2296.6-1.1;
objectX = -5239.3-1.1; objectY = 2292-1.1;

middle = plot(x,-z,'Color',[0 0 1]);
hold on;
lSide = plot(x,-z-1, 'Color', [0.8 0.8 1]);
rSide = plot(x,-z+1, 'Color', [0.8 0.8 1]);
axis equal;
axis ([objectX-60 objectX+40 objectY-6 objectY+15]);
yL = get(gca,'YLim'); \% length of Y
set(gca,'xtick',[],'ytick',[])
steeringStarted = line([x(steerDiff(1)) x(steerDiff(1))],yL,
'Color',[0 1 0]);
\% drive line, where the plotting started

objectShown = line([objectX-35 objectX-35],yL,'Color',[1 0 0]);
\% line where object appeared

legend('Middle of the car', 'Border of the car',
'Border of the car', 'Start of steering','Object was shown' );

rectangle('Position',[objectX objectY 2.2 2.2]); \%5400.7, 1.1, 2296.6
\%rectangle('Position',[-5238.2 2290.9 2.2 2.2]); \%5239.3, 1.1, 2292
reactionTime = x(steerDiff(1))- (-5399.6-35);
hold off;

```

D.2 Results of Lane Change Test

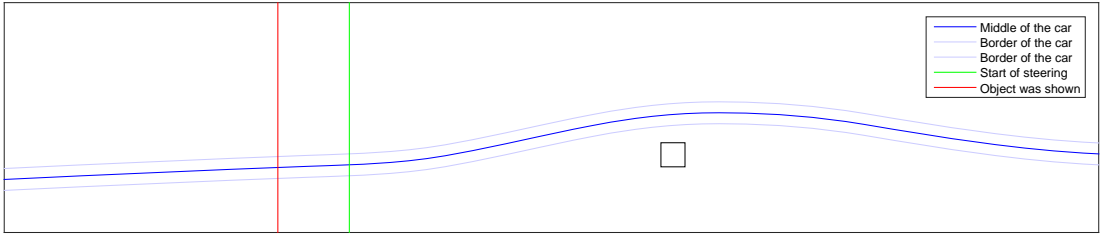


Figure D.1. Lane Change Test - Participant #1

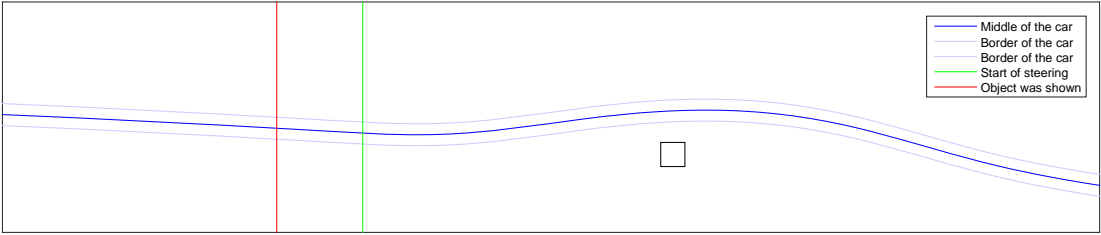


Figure D.2. Lane Change Test - Participant #2

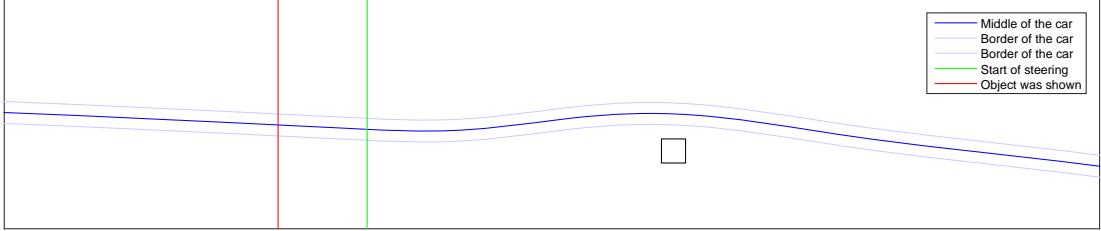


Figure D.3. Lane Change Test - Participant #3

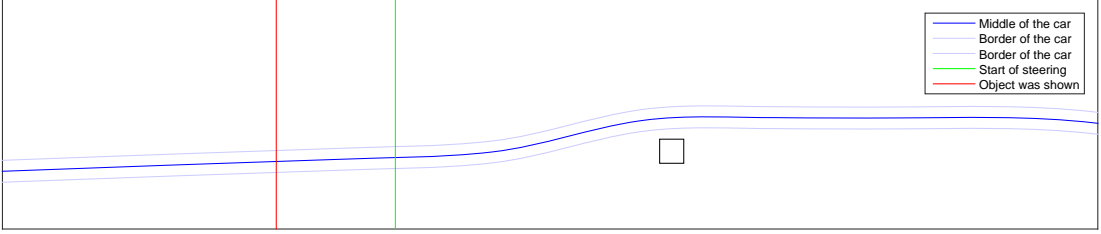


Figure D.4. Lane Change Test - Participant #4

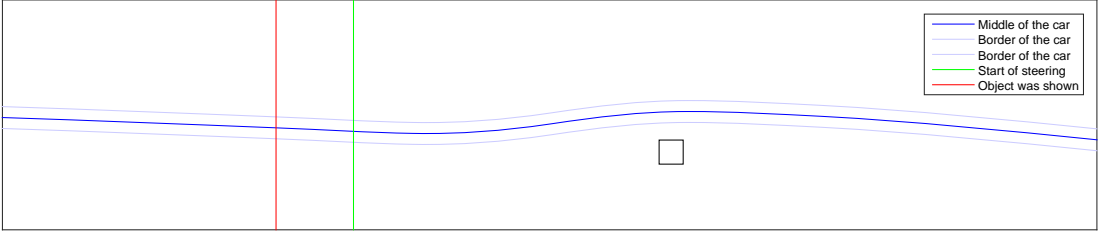


Figure D.5. Lane Change Test - Participant #5

Appendix E

Content of the CD

```
|---logging-server
|   |---wa2car
|       |---.google_apis
|       |---src
|           |---cz
|           |   |---lukashruby
|           |       |---wa2car
|           |           |---client
|           |           |---server
|           |           |---shared
|           |---META-INF
|---mobile-application
|   |---CarDashboardCore
|       |---src
|           |---main
|               |---java
|                   |---com
|                       |---eclubprague
|                           |---cardashboard
|                               |---core
|                                   |---adapters
|                                   |---application
|                                       |---eventbus
|                                       |---update
|---data
|   |---database
|   |---modules
|---fragments
|---model
|   |---eventbus
|       |---events
|       |---interfaces
|---interfaces
|---resources
|---validation
|---modules
|   |---base
|       |---models
|       |---custom
|           |---settings
|           |---predefined
|---obd
|---preferences
```



```
| | |---main
| | |   |---java
| | |     |---cz
| | |       |---blahami2
| | |         | |---cardashboardadapter
| | |         |---lukashruby
| | |           |---fel
| | |   |---test
| | |     |---java
| | |       |---cz
| | |         |---lukashruby
| | |           |---fel
| | |---target
| | |   |---maven-archiver
| | |   |---maven-status
| | |     |---maven-compiler-plugin
| | |       |---compile
| | |         | |---default-compile
| | |         |---testCompile
| | |           |---default-testCompile
| | |   |---surefire-reports
|---thesis
|   |---doc
|   |---resources
|     |---images
|       |---mockups
|       |---screenshots
|       |---tests
|       |---ui
|---template
```