Vítor Emanuel Ornelas Gomes

Deteção de Acidentes e Implementação de eCall Baseada em Smartphone

Smartphone Based Accident Detection and eCall Implementation

Vítor Emanuel Ornelas Gomes

Deteção de Acidentes e Implementação de eCall Baseada em Smartphone

Smartphone Based Accident Detection and eCall Implementation

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requesitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Doutor Arnaldo Silva Rodrigues de Oliveira (orientador), Professor Auxiliar do Departamento de Electrónica Telecomunicações e Informática da Universidade de Aveiro e do Doutor Joaquim José de Castro Ferreira (coorientador), Professor Adjunto da Escola Superior de Tecnologia e Gestão de Águeda da Universidade de Aveiro.

o júri / the jury

presidente / president

Prof. Doutor João Nuno Pimentel da Silva Matos
Professor Associado do Departamento de Engenharia Electrónica Telecomunicações e Informática da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor José Carlos Meireles Monteiro Metrôlho
Professor Adjunto do Departamento de Engenharia Informática da Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco

Prof. Doutor Arnaldo Silva Rodrigues de Oliveira
Professor Auxiliar do Departamento de Engenharia Electrónica Telecomunicações e Informática da Universidade de Aveiro

agradecimentos / acknowledgements

A realização desta dissertação é o cultimar de vários anos de aprendizagem, levados a cabo na magnífica Universidade de Aveiro, à qual agradeço a formação de exelente qualidade que me ofereceu.
Além da formação académica, este anos de vivências, distante de casa, transformaram-me na pessoa que sou hoje. Quero agradecer a todos os meus colegas que contribuiram para estes maravilhosos anos de estudante.
Quero agradecer também ao Doutor Arnaldo Oliveira e ao Doutor Joaquim Ferreira, assim como a toda a equipa de insvestigação do HEADWAY por toda a ajuda e orientação prestada ao logo deste projecto.
Por fim quero agradecer à minha namorada e aos meus pais e irmão pelo apoio e motivação incondicionais que me deram ao longo deste meu percurso académico.

This dissertation is the result of several years as a student at the magnificent University of Aveiro, to whom I'm thankful for the excellent education quality it provided me.
Moreover, the years I lived away from home, provided me experiences that made me the person I am today. I want to thank all my colleagues that contributed to these wonderful student years.
I also want to thank to professors, Arnaldo Oliveira and Joaquim Ferreira, as well as to all HEADWAY's research team, for all the help and guidance they provided me throughout this project.
Finally, I want to thank my girlfriend, my parents and by brother for the unconditional support and motivation provided to me during my student years.

Palavras-chave

Android, Detecção Automática de Acidentes, Comunicações Veículares, Sistemas de Transporte Inteligentes, eCall.

Resumo

Os Sistemas de Transporte Inteligentes estão a emergir, de forma a introduzir mais segurança, eficiência e conforto nas estradas.
Esta inteligência deve-se ao facto de novas tecnologias estarem a ser introduzidas nos automóveis recentes.
Como resultado da evolução tecnológica os sistemas de comunicação veículares estão a ser desenvolvidos, com o objectivo de munir os condutores com informações relativas aos diferentes intervenientes da estrada onde circulam. Prevê-se que este tipo de informação leve a uma maior segurança e eficiência nas estradas.
Actualmente no Instituto de Telecomunicações de Aveiro, está a decorrer um projecto que visa fornecer uma alternativa como sistema de comunicações veículares. Este projecto tem o nome de HEADWAY. O HEADWAY é um sistema de comunicações veículares DSRC 5.9 GHz, atualmente em desenvolvimento.
Os smartphones hoje em dia já são dispositivos estabelecidos no mercado. Isto deve-se ao facto destes apresentarem um grande potencial, ao integrarem recursos de hardware incríveis num pequeno dispositivo e de permitirem o desenvolvimento de aplicações por terceiros. A creatividade dos programadores tem permitido a ultilização destes dispositivos em diversas áreas.
De forma a diminimuir o numero de mortes causadas por acidentes rodoviários, a Comissão Europeia, tornou obrigatório que em 2015 todos os novos carros estejam equipados com o sistema eCall, que visa a detecção de acidentes e pedido de ajuda ao 112 automáticos.
Esta dissertação tem por um lado, o objectivo de desenvolver um detector de acidentes com implementação de eCall, e, por outro lado, integrar um smartphone com o HEADWAY, através do desenvolvimento de uma aplicação que tire partido das caracteristicas deste sistema e assim o demonstre.
Para cumprir os objectivos foi desenvolvida uma aplicação para Android que atua como HMI para o HEADWAY, facilita a troca de mensagens entre veículos, detecta automaticamente acidentes e procede com pedidos de ajuda.
Na conclusão do projecto, verificou-se que os objectivos propostos foram na sua maioria concluídos, exceptuando a implementação da eCall ao 112, sendo desenvolvido um método alternativo.

Keywords Android, Automatic Accident Detection, Vehicular Communications, Intelligent Transportation Sistemas, eCall.

Abstract

Intelligent Transportation Systems are emerging, to increase safety, efficiency and comfort on roads.
This intelligence is due to the fact that new technologies are being introduced in the most recent automobiles.
As a result of this technological evolution, vehicular communication systems are being developed, to provide drivers with more information about the interventionists present in the roads they circulate. Predictions point that this information can increase safety and efficiency on roads.
Presently, the *Instituto de Telecomunicações de Aveiro*, is developing its own vehicular communication system, named HEADWAY, as a solution. HEADWAY DSRC 5.9 GHz vehicular communication system currently under development.
Smartphones nowadays are very popular devices. This is due to the fact that they pack incredible hardware resources in a small and portable device and the possibility to third party developers, develop applications for them. This enables these devices to be used in different areas, depending only from the creativity of the developers.
To diminish the number of fatalities due to road accidents, the European Commission has mandated the implementation of eCall in every new vehicle by 2015. In vehicles, the eCall aim to detect accidents and request accidents automatically.
This dissertation targets, on the one hand, the development of an accident detection mechanism with eCall implementation. On the other hand it targets the integration of smartphones with HEADWAY, by developing an application that takes advantage of the system characteristics and demonstrates it.
To achieve the proposed goals, an Android application was developed which acts as an HMI for HEADWAY, enables message exchange between vehicles, automatically detects accidents and proceeds with a help request.
Most of the proposed goals where achieved, except the eCall implementation, which an alternative method was developed.

Contents

List of Figures

List of Tables

Acronyms

AAD Autonomous Accident Detection ix, 2, 3, 5, 21–24, 29, 31, 33–35, 41, 44, 59, 67

ADA Accident Detection Algorithm ix, 35, 36, 41, 43–45, 48, 52, 53, 55–59, 61, 65, 67, 74, 75

ADC Analog to Digital Converter ix, 32

ADT Android Development Tools ix, 18

AIS Abbreviation Injury Scale ix, 24

AOSP Android Open Source Project ix, 18

API Application Programming Interface ix, 17–19, 21, 60, 62, 75

ASI Accident Severity Index ix, 24, 25, 60, 62, 67–70

AU Application Unit ix, 7, 31, 33–35, 40, 41, 44, 75

BSA Basis Set of Applications ix, 13

BSD Berkeley Software Distribution ix, 19

BSM Basic Safety Messages ix, 8

C2C-CC Car to Car Communication Consortium ix, 7

CAM Cooperative Awareness Message ix, 9, 11, 13, 33, 75

CCH Control CHannel ix, 11

CLI Caller Line Identification ix, 29

DAC Digital to Analog Converter ix, 32

DENM Decentralized Environmental Notification Message ix, 9, 11, 13, 33–36, 41, 75

DSRC Dedicated Short Range Communications ix, 9, 32

ECU Electronic Control Unit ix

EG eCall Generator ix, 28, 29

EMS Emergency Medical Systems ix, 27, 31, 62

ESP Electronic Stability Program ix, 15

ETSI European Telecommunications Standards Institute ix, 2, 3, 8, 9, 12, 31, 33, 75

EU European Union ix, 2, 6, 27

FPGA Field-Programmable Gate Array ix, 32

GLONASS GLObal NAvigation Satellite System ix, 1

GPS Global Positioning System ix, 1, 2, 18, 23, 36, 44, 48, 57

GSM Global System for Mobile Communications ix, 1, 8, 16, 52

GUI Graphical User Interface ix, 3, 32–36, 41, 45, 56, 65

HEADWAY Highway Environment ADvanced WArning SYstem ix, 2, 3, 5, 9–12, 29, 31–34, 41, 49, 75

HIC Head Injury Criteria ix, 24

HMI Human to Machine Interface ix, 10, 11, 13, 33, 34, 41

I2V Infrastructure to Vehicle ix, 8

IDC Internet Data Corporation ix, 16

IDE Integrated Development Environment ix, 18, 19

IEEE Institute of Electrical and Electronics Engineers ix, 2, 8, 11, 31

IP Internet Protocol ix, 28

IPv4 Internet Protocol version 4 ix, 28

IPv5 Internet Protocol version 5 ix

ISEL Instituto Superior de Engenharia de Lisboa ix, 2, 9

IT-Aveiro Instituto de Telecomunicações de Aveiro ix, 2, 9, 23

ITS Intelligent Transportation Systems ix, 2, 3, 5–7, 9, 11–13, 15, 21, 23, 29, 34, 75

LTE Long Term Evolution ix, 1, 8, 16

MAC Media Access Control ix, 11, 12, 32

MIB Management Information Base ix, 11

MLME MAC Layer Management Entity ix, 11

MNO Mobile Network Operator ix, 27, 29, 73

MSD Minimum Set of Data ix, 27–29, 31, 62

NDK Native Development Kit ix, 18

NFC Near Field Communications ix, 1, 16, 18

OBD-II On-Board Diagnostic version 2 ix, 23, 32, 35, 36, 40, 41, 44, 45, 49, 52, 60, 62, 67

OBU On Board Unit ix, 7, 8, 11, 12, 31, 34–36, 41, 60, 74, 75

OFDM Orthogonal Frequency-Division Multiplexing ix, 11

OHA Open Handset Alliance ix, 18

OS Operating System ix, 16, 17, 19, 20, 43, 65, 75

PDU Protocol Data Unit ix, 14

PHY PHYsical ix, 11, 12, 32

PID Parameter Identificator ix, 44

PSAP Public Safety Answering Point ix, 27–29

RF Radio Frequency ix, 32

RHW Road Hazard Warning ix, 3, 13, 31, 33–37, 40, 41, 44, 45, 49, 52, 53, 55, 57, 58, 62, 75

RMA Resource Manager Application ix, 10

RSU Road Side Unit ix, 7, 8, 12, 31

SAE Society of Automotive Engineers ix, 8

SCH Service CHannel ix, 11

SDK Software Development Kit ix, 18, 19

SMS Short Message Service ix, 23, 53, 62, 73

THIV Theoretical Head Impact Velocity ix, 25

UI User Interface ix, 31, 37, 56, 84

UML Unified Modeling Language ix, 51

UMTS Universal Mobile Telecommunications System ix, 1, 8, 16

URL Uniform Resource Locator ix, 20

USB Universal Serial Bus ix, 16, 20, 32, 44, 45, 53, 61, 62, 65, 74

UTC Coordinated Universal Time ix, 14, 28, 67–69

UX User Experience ix, 18–21, 34

V2I Vehicle to Infrastructure ix, 8

V2V Vehicle to Vehicle ix, 8, 23

V2X Vehicle to Anything ix, 8–10, 12, 13, 29, 33

VANET Vehicular Ad-hoc NETwork ix, 7, 29, 31

VCR Videocassette Recorder ix, 1

VIN Vehicle Identification Number ix, 28, 62

VM Virtual Machine ix, 18

WAVE Wireless Access in Vehicular Environments ix, 10, 31

WME WAVE Management Entity ix, 11

WSMP WAVE Short Message Protocol ix, 11

Chapter 1

Introduction

1.1 Motivation and Context

The Human needs for comfort and safety demands the use of their creativity to find solutions to the existing problems. Ideas grow and evolve with the contribution of different people.

Technology has been evolving throughout the years to answer the needs of the Human population in several areas, from heath to entertainment. Technology, usually, is firstly developed to provide alternatives to a certain situation, but it ends up evolving to areas where its creators never thought it would.

Throughout the years, we've seen technological products being developed to complement or take advantage of other products, like the Videocassette Recorder (VCR) did with the TV sets. This combination of technological products is present everywhere and the Human creativity is the limit.

A recent technological trend is the technology integration, i.e, the aggregation of several technological products into a single one, reducing the number of devices needed in a day-to-day life. The perfect example of that is the smartphone.

Devices such as smartphones are becoming increasingly popular. Today, smartphones are usually equipped with dual-core processors, big screens, sensors (accelerometer, gyroscope, magnetometer, etc.), location systems such as Global Positioning System (GPS) and GLObal NAvigation Satellite System (GLONASS), powerful cameras and different communication systems such as Global System for Mobile Communications (GSM), Universal Mobile Telecommunications System (UMTS), Long Term Evolution (LTE), Wi-Fi, Bluetooth and Near Field Communications (NFC). If the success of these devices is, on the one hand, due to its powerful hardware integrated in a small and portable device, on the other hand is also due to the mobile platforms running on these devices which provide a powerful and simple experience to users, allowing smartphones to be used by anyone.

Due to its powerful hardware and software, smartphones are being used for different purposes such as communications, entertainment, social networking, internet browsing, vehicle navigation, photo shooting, productivity, personal tasks assistant, among other uses. Most mobile platforms provide third party developers, the necessary tools for applications development, extending the use of smartphones to the their creativity. Applications are stored and organised in repositories that provide an easy way for users to get them and for developers to expose their work. Here applications can be purchased or installed free of charge by users

in their devices.

Many services like transportation systems, banking, e-commerce and many others have developed applications for smartphones in order to deliver their products comfortably to their costumers.

Transportation systems, particularly automobiles, are the focus of this dissertation. The automobile has been evolving quickly throughout the years, due to its enormous and valuable industry. At the moment, this evolution has resulted in introducing vehicles with greater safety, efficiency and comfort. The auto-makers, to deliver better automobiles, traditionally invest and research in materials science, breaking systems, aerodynamic designs, engine efficiency increase and new automation technologies that provide more comfort to the driver.

Despite the progress that the auto-maker industry achieved in producing safer and more efficient vehicles over the last years, road accidents were still responsible in European Union (EU) for 28000 fatalities in 2012 and this is the lowest number since the last 10 years [1]. The amount of carbon emissions released by vehicles worldwide is also an important and global concern.

To address these concerns, new technological capabilities are being introduced in automobiles. This is the case of vehicular communication systems. With the vehicular communication systems development, the Intelligent Transportation Systems (ITS) concept is emerging. ITS aim to increase road safety and travel efficiency. Vehicular communications are at the moment a strong research topic in the communications scientific community. A great number of universities, institutes, auto-makers and telecommunication companies are researching and developing solutions to be deployed at a large scale. The need for standardization is a concern, in order to unite and direct the interventionist efforts. The European Telecommunications Standards Institute (ETSI) [2] and the Institute of Electrical and Electronics Engineers (IEEE) [3] already published standards to be followed in this application field.

The project Highway Environment ADvanced WArning SYstem (HEADWAY) [4] is a vehicular communications system under development, that follows the ETSI and the IEEE standards for vehicular communications. This project aim is to provide a solution for these type of communications. HEADWAY is a result from a partnership between *Brisa Inovação* (Portuguese motorway concessionaire) and the research centres *Instituto de Telecomunicações de Aveiro (IT-Aveiro)* and *Instituto Superior de Engenharia de Lisboa (ISEL).*

Its expected that the communication between vehicles is able to provide drivers with more information about their surroundings, allowing them to make better decisions, resulting in the increase of their safety and efficiency. With more information drivers can decide the best route to take, or even carefully approach an unsafe location.

High end automobiles, today, offer some ITS services built-in on vehicle's on-board computers, such as turn-by-turn GPS navigation systems, accident detection system, as well as traffic, weather and entertainment applications. On older and lower end automobiles, smartphones are already being used, to bring those same features and services. Smartphones nowadays are a valuable solution to push ITS into older vehicles.

The context of this dissertation is to develop a smartphone mobile application that takes advantage of the device's hardware resources and implements vehicle eCall system composed by a Autonomous Accident Detection (AAD) mechanism along with an help request mechanism (described in section 2.6). This application should be integrated with HEADWAY in order to take advantage and to demonstrate it.

1.2 Goals

This dissertation goal is, on the one hand, to develop an eCall system composed by AAD mechanism based on a smartphone, to detect road accidents and perform an help request based on the eCall requirements. On the other hand, a smartphone integration with HEADWAY vehicular communications system is proposed, in order to demonstrate its potential.

The dissertation targets the development of a smartphone application named HDy Copilot (HEADWAY Copilot). The application should be able to detect accidents, trigger an eCall help request, take advantage of HEADWAY to exchange road safety and efficiency related messages with other vehicles and provide an interface to allow interaction with HEADWAY. With HDy Copilot, the smartphone should be integrated with the vehicular communication system, take advantage and demonstrate its capabilities. HDy Copilot should provide the following features:

- Graphical User Interface (GUI) - The application should provide the means to interact with HEADWAY, by generating a GUI. This GUI should be designed according with the context of use of the application (driving act).
- Road Hazard Warning (RHW) report - To allow the user to send a location and time based messages notifying road related events such as traffic congestion, road hazards (oil, rocks, animals, etc...) and stationary vehicles (involved in accidents).
- eCall system - Use smartphone hardware resources such as sensors to detect car accidents. As a response to the accident detection, the application should automatically broadcast an accident notification to other users throughout HEADWAY and trigger an eCall help response (described in 2.6).

RHW are exchanged in vehicular systems that implement ETSI protocol stack (described in 2.1.2).

1.3 Dissertation structure

This document is divided into the six following chapters:

- **Chapter 1 - Introduction** - Contextual introduction, along with the dissertation goals. In this chapter the vehicular communication project HEADWAY and the smartphone technologies are firstly introduced.
- **Chapter 2 - Background and related work** - Research that supports the dissertation. The ITS and HEADWAY state of the art is presented, followed by the study and comparison of the most used smartphone mobile platforms. The state of the art of smartphone usage in ITS is also presented with some examples of commercial applications available for the smartphone users as well as ongoing research projects, from the scientific community, in this area. As one of the main goals HDy Copilot is the eCall system, both the accident detection and the eCall subjects performed research is presented at the end of the chapter.
- **Chapter 3 - Proposal** - Presents a congregation of the research performed in chapter two, with the dissertation goals and is presented the drafts and guidelines of a solution to be implemented and tested.

- **Chapter 4 - Implementation** - Describes the HDy Copilot implementation. Here all the relevant aspects of the application implementation are described.
- **Chapter 5 - Tests and validation** - Presents performed tests results to assess the HDy Copilot implemented functionalities validity.
- **Chapter 6 - Conclusion -** Presents the dissertation conclusion and possible future improvements to HDy Copilot and its integration.

Chapter 2

Background and related work

This chapter presents all the background research performed to sustain this dissertation.

First it introduces the ITS concept and the HEADWAY project. The goals and a general overview of HEADWAY implementation and context in ITS are described.

In a second section the market penetration and technical characteristics of the relevant mobile platforms are described. An evaluation and comparison is performed in order to assess the suitable platform for HDy Copilot development.

Later on the chapter, related projects and applications are presented as a result of a search. These researched projects provide a starting point for the AAD. A search is also performed in the AAD field to assess what can be implemented in the smartphone for that purpose.

Finally the eCall project is described to assess its features and functionality requirements in order to implement it in HDy Copilot.

2.1 Intelligent Transportation Systems

Transport of people and goods go back a long time in Human history. The expansion of empires and the need for trade, started the demand for shorter travelling times. Transportation started with the use of animals such as horses, camels and oxen. Later in history water navigation transports emerged, providing the means to transport people and goods overseas, with the use of the sail boats. This was a significant advancement in transportation technologies. Later on, with the invention of the steam engine, new transportation systems emerged, such as trains, automobiles and steam engine boats. These transportation systems all relied on an internal power source to enable movement, the steam engine, instead of using animals or wind.

With the steam engine came the trains and the automobiles, which led to the construction of transport infrastructure such as train lines and roads.

More recently the internal combustion engine was introduced and brought more efficient and smaller engines. With these, transportation started to become cheaper and therefore more people were able to use it.

Today transportation systems are widely integrated with our day to day life. As the transportation systems became faster, more powerful and widely used, the safety risks involved increased. Fatalities due to transportation, especially in automobiles, are a global concern. Another related global concern is the environmental pollution. The internal combustion engine is the most used in automobiles. Due to the type of fuel used to generate power,

it releases toxic gases such as CO_2 into the atmosphere, contributing to the increase of the green house effect. These toxic gases are also responsible to inflict respiratory problems and harming people's health.

Transportation systems affect directly people quality of life. Several important aspects of peoples lives are connected in different ways to transportation systems. Health, economy and time spent in daily routes are the best examples. The air pollution generated by vehicles, severely affects people's health. The economic and time cost of transportation affects family budgets and quality of life. Making transportation systems more efficient can therefore be directly correlated to improving people's lives [5].

Transportation system have been evolving towards efficiency and safety increase and with it the ITS concept is emerging. This evolution resulted in the development of vehicular communications systems, which allow a real time exchange of information.

2.1.1 Main benefits of ITS

Transportation safety, pollution reduction and time/cost efficiency are some of the most important goals in ITS. This concept aims to bring benefits to people's lives. According to [5] the benefits can be grouped in the following three categories:

- **Safety increase** - Vehicle manufacturers are investing resources in researching areas like new materials to diminish the impact acceleration, in order to protect vehicle occupants, technologies to assist drivers such as navigation systems, braking systems and collision avoidance systems. Some manufactures also provide services to detected impacts and request automatic assistance in order to mitigate severe damages caused when an accident occurs. Starting in 2015, all new cars in EU will have integrated an eCall system (described in 2.6).

  Making use of vehicular communications, its possible to keep drivers informed and make them aware of the possible dangers in their route, allowing for a better decision making.

- **Environment preservation** - When a driver uses longer routes, circles an area looking for a parking space or is stuck in a traffic jam, the vehicle consumes more fuel and therefore pollutes more. Bringing navigation systems and real-time information to the vehicle, benefits these situations and helps to reduce CO_2 emissions. With early information, a driver can opt for an alternative route in case of a traffic jam detection or navigate directly to a parking space in case he is looking for one.

- **Transport efficiency** With vehicular communications, its possible to exchange informations regarding a route and predict the amount of time spent in travelling that route. If vehicle location, weather forecast, traffic conditions and other route conditions are available, a more efficient route planing is possible. For goods transportation companies this can be translated in a better and more cost efficient service. Transportation efficiency is directly connected with environment preservation as well as time and cost reduction.

  A popular ITS application in Portugal is the electronic tool payment called *Via Verde* [6]. It allows for a more efficient payment method and therefore a more efficient traffic flow in motorways.

Some of the technologies mentioned in 2.1.1 such as vehicular communications are still under development and are not yet deployed in vehicles.

2.1.2 Vehicular communication network

The next logical step for ITS is allowing vehicles to be aware of road intervening elements. Knowing the exact position, velocity and acceleration of other nearby vehicles as well as accessing information about what to expect down on the route, allow for a better planning and decision making, thus contributing for safety and efficiency increase.

Vehicular communications, due to the vehicle high speed mobile environment, are different from the other type of networks. To enable this communication, the concept of Vehicular Ad-hoc NETwork (VANET) emerged. These networks are spontaneously formed between moving vehicles with wireless communication interfaces. Vehicular networks are still a work in progress and there are research projects under development that produce new ideas which can update the VANET concept. An architecture reference for vehicular networks is provided the Car to Car Communication Consortium (C2C-CC) and published by Zhang et al. in [7]. Figure 2.1 depicts that architecture.

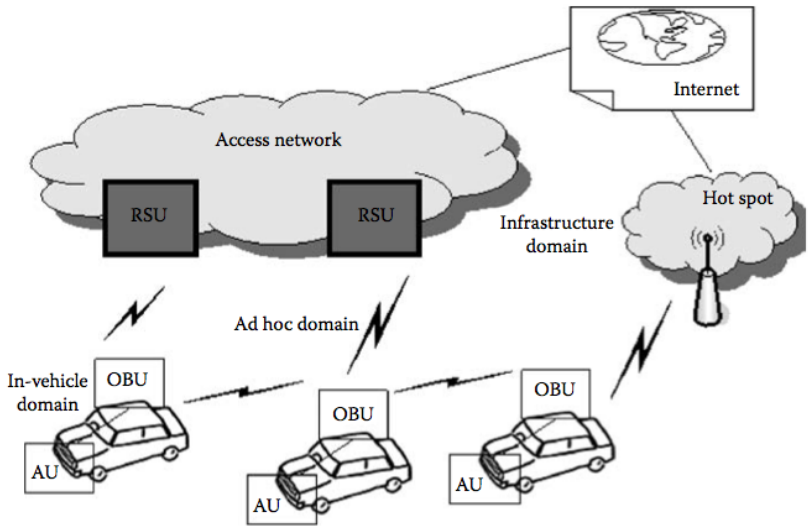


Figure 2.1: C2C-CC vehicular networks reference architecture. *Source: [7]*

This architecture reference distinguishes three domains. The in-vehicle, the ad-hoc and the infrastructure domain. The in-vehicle domain is composed by two units: the On Board Unit (OBU) and the Application Unit (AU). The OBU is a device with communication capabilities and the AU is a device executing applications that take advantage of OBU's communication capabilities. The AU and OBU distinction is logical, as they can be integrated in the same physical device. The AU can also be a separate device that can be dynamically attached or detached to/from the OBU. The communication between these two units can be wired or wireless. The ad-hoc domain is where VANET concept is integrated. This domain is composed by two units, the already known OBU and the Road Side Unit (RSU). RSUs are stationary along the road and can be attached to an infrastructure network, which can be

connected to the internet. RSUs can communicate with each other and with OBUs. The ad-hoc concept implies that every OBU and RSU can behave as a network node, receiving or redirecting information throughout the network. The infrastructure domain can be composed by RSU's and hot spots. RSUs allow the OBU to communicate with Internet. The hot spot also facilitates this communication using technologies such as GSM, UMTS, WiMax and LTE, if they are integrated in the OBU.

Two communication directions come out of these networks. Vehicle to Vehicle (V2V) and Infrastructure to Vehicle (I2V) (or Vehicle to Infrastructure (V2I)). Usually a general designation that integrate both communication directions is Vehicle to Anything (V2X). This designation is used throughout this document to refer to vehicular communications.

These networks are a cause of great interest by the scientific community and the involved industries. The projects that are being develop are resulting in standards that define specifications to be followed.

There are two main protocol stacks for V2X communications, differentiated by the standards implemented. These stacks are responsibility of two standard organizations, ETSI [2] (Europe) and IEEE [3] (USA). Figure 2.2 depicts the protocol stack from these two organizations.

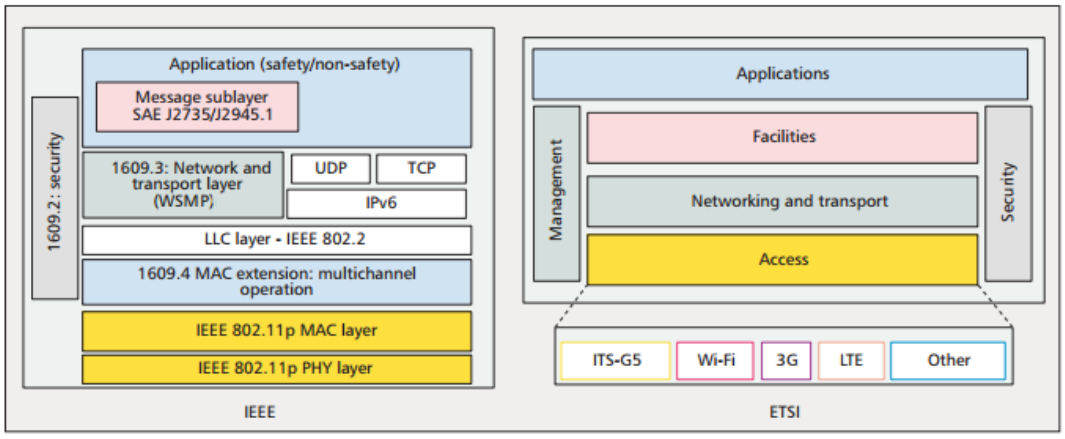


Figure 2.2: IEEE and ETSI vehicular communication protocol stack *Source: [8]*

As figure 2.2 depicts the architecture followed by the European and the American protocol stacks share some design choices. These protocol stacks are still ongoing projects and can be changed. The main differences between these two approaches are depicted in table 2.1.

The differences between the two are mainly in the radio spectrum allocation and access as well as the main safety messages. The ETSI stack predicts the use of different wireless technologies such as Wi-Fi, UMTS, LTE, ITS-G5 (ETSI vehicular communication standard) among others to establish communications. For this dissertation, the most important differences between this two protocol stacks are the main safety messages. In IEEE these messages are implemented in the Message sublayer using the Society of Automotive Engineers (SAE) J2735/J2945.1 standards and are designated Basic Safety Messages (BSM). In the ETSI stack this messages are implemented in the Facilities layer. There are two types of safety messages

	United States	European Union
Total allocated spectrum	75 MHz	50 MHz
Frequency range	5.85-5.925 GHz	5.85-5.905 GHz
Control Channel frequency	5.89 GHz (CH178)	5.9 GHz (CH180)
Number of Service Channels (SCHs)	6 (CH172, CH174, CH176, CH180, CH182, CH184)	4 (CH172, CH174, CH176, CH178)
Main standardization bodies	IEEE, SAE International, FCC	ETSI, ISO/CEN, CEPT
PHY/MAC layers	IEEE 802.11p, 1609.4	Multi access technologies, among which ETSI ITS-G5, same as 802.11p
Higher layers	IEEE 1609.0, 1609.2, 1609.3	ETSI ITS Station: Facility, Networking and Transport, Security layers
Applications / message formats	IEEE 1609.x, SAE International	ETSI
Service announcements	WSA (WAVE Service Advertisement)	SAM (Service Announcement)
Main safety messages	BSM (Basic Safety Message)	CAM, DENM

Table 2.1: Multichannel VANET standards: United States vs. Europe. *Adapted from: [8]*

in ETSI protocol stack, the Cooperative Awareness Message (CAM) and the Decentralized Environmental Notification Message (DENM). These messages structure and purpose is described in section 2.2.3.

2.2 HEADWAY

One key technology to allow the development of ITS is wireless vehicular communications. This feature will allow the development of the services needed to bring the benefits described in section 2.1.1. Today, some standards are under development to set the rules and guidelines to be followed by researchers in the vehicle communication systems field. This section introduces the vehicular communication project in which this dissertation is integrated, its context in ITS and goals.

2.2.1 Project definition

HEADWAY is a project led by *Brisa Inovação* [4] in partnership with IT-Aveiro and ISEL. It aims to develop a V2X Dedicated Short Range Communications (DSRC) system

in the 5.9 GHz band, that is currently allocated by the European Commission for vehicular communications [9].

The goal of the project is to provide services to motorway drivers in the range of safety, comfort and infotainment in order to reduce the number of accidents and provide commodity of travelling while source of revenue to motorway and/or telecommunication operators.

According to [4], HEADWAY main applications and services guidelines are the following:

- **Warning location notification** - Use Human to Machine Interface (HMI) to warn drivers against upcoming accidents, incidents, weather and route conditions that are detected by other vehicles or roadside telematic systems and transmitted through V2X communication systems.
- **Incident/traffic jam/road works warning** - Disseminate information about warnings, incidents and dynamic events over the network, such as variable speed limits.
- **Decentralized floating car data collection** - Using vehicles as positioning devices and sensing units to gather and disseminate road network data for socially optimal routes.
- **Route guidance support** - Alternative route recommendation to direct drivers around congested locations and traffic load distribution.
- **Interoperable tolling system** - Universal and interoperable tooling system.
- **Access control/Parking assistance** - Access control. Grants access to restricted areas automatically, for example, environmental zones in cities.
- **In-vehicle signage/HMI support** - Use HMI to display directions and warning information

2.2.2 Implementation structure

The implementation structure of HEADWAY contains parts from the IEEE protocol stack and parts from the ETSI protocol stack. This two protocol stacks are not completely disjointed. Some ETSI standards refer Wireless Access in Vehicular Environments (WAVE) standards. HEADWAY follows the mentioned protocol stacks and as a result its implementation structure is layer based.

WAVE standards

The IEEE protocol stack is based on the WAVE standards. Not all of them are part of the HEADWAY implementation structure. According to [9] the standards implemented are the following:

- IEEE P1609.0 - Draft Standard for WAVE Describes WAVE architecture and services necessary for multi channel DSRC/WAVE devices for vehicular environment.
- IEEE 1609.1 - Trial Use Standard for WAVE - Specifies the Resource Manager Application (RMA) services, interfaces and data formats for applications to communicate between architecture components.

- IEEE 1609.2 - Trial Use Standard for WAVE - Defines the security services for applications and messages encryption in vehicular environments.
- IEEE 1609.3 - Trial Use Standard for WAVE - Defines network and transport layer services. It also define the WAVE Short Message Protocol (WSMP) and the Management Information Base (MIB) for the WAVE protocol stack. The WSMP is a network protocol developed for real-time communication.
- IEEE 1609.4 - Trial Use Standard for WAVE - Multi-Channel Operations and enhancements to the IEEE 802.11 Media Access Control (MAC) to support WAVE operations
- IEEE P1609.11 - Over-the-Air Data Exchange Protocol for ITS - It will define the services and secure message formats necessary for electronic tool payments
- IEEE Std 802.11p - This standard is an amendment to the IEEE 802.11 which defines rules and specifications for wireless local communications in 2.4 GHz, 3.6 GHz and 5 GHz, where the vehicular communication environment stands. The IEEE 802.11p defines parameters such as transmit power limits, channel spacing and the frequency bands that may be used in each location.

Layers

In the most recent publication about HEADWAY [9], the implementation structure presents the Application Layer, Network Layer, MAC Layer, a PHY layer.

The Application layer is based on the IEEE 1609.1 standard. Its responsible for managing resources through the RMA and interacting with the OBU's resource command processor.

The Network layer is based on IEEE 1609.3 and defines the network transportation services, including the addressing and data routing required for data exchange between WAVE entities. It has a management plane and a data plane, where the first is composed by the WAVE Management Entity (WME), responsible for the MIB and advertising services. The data plane implements the IPv6 and WSMP protocols for short message exchange.

The MAC layer is based on IEEE 802.11p and IEEE 1609.4 and its composed by a management plane named MAC Layer Management Entity (MLME) and a data plane. The MLME is responsible for coordinating regular switching between Control CHannel (CCH) and Service CHannel (SCH) and queues service advertisements. The data plane is responsible for IPv6 and WSMP message queuing and transmission on the correct channel. The MAC layer is responsible for coordinating medium access.

The PHYsical (PHY) layer is based on the IEEE 802.11p specifications for Orthogonal Frequency-Division Multiplexing (OFDM). Its responsible for data processing at the physical level such as encoding/decoding, interleaving/deinterleaving as well as modulation/demodulation operations.

As HEADWAY is an ongoing project, other layers are being implemented that [9] does not present. The Security layer is based on IEEE 1609.2 standard. Its responsible for security issues related to message encryption and validation.

Another new layer implemented is the Facilities layer, from ETSI protocol stack. This layer is responsible for supporting a HMI for the driver, constant possession of location and time reference data, managing and crossing information from different sources, generating, routing and transmitting CAM and DENM messages. This messages are described with more detail in 2.2.3.

The protocol layers require different technology resources for their implementation. The Application, Network, Facilities, Security and upper MAC can be implemented in software, because their performance requirements are less intensive. The lower MAC and PHY layers are computationally intense, because of time critical functionalities such as synchronization, channel coordination and bit oriented operations, therefore are more efficiently implemented in hardware. The analogue PHY layer also contains a transceiver and an amplifier. Figure 2.3 depicts the current implementation structure of HEADWAY.

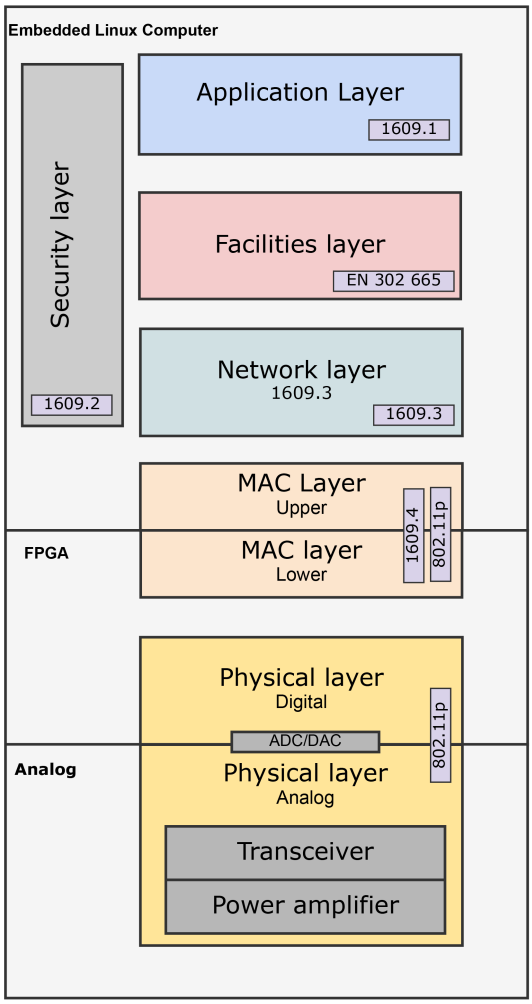


Figure 2.3: HEADWAY implementation structure. *Adapted from [9] and [10]*

2.2.3 CAM and DENM messages

Vehicular networks provide vehicles with the ability to exchange data between themselves. Data is exchanged in predefined formatted messages in order to be successfully interpreted by the different ITS OBUs and RSUs that participate in the V2X network. HEADWAY implements the ETSI Facilities layer, where these messages are generated and managed.

In the Facilities layer two types of messages can be generated. The CAM messages and the DENM messages. The broadcasting methods of these messages are different and they suit different purposes.

Cooperative Awareness Message (CAM)

According with the ETSI TS 102 637-2 [11], CAMs provide information of presence, position and basic status of near by ITS station (RSUs and OBUs), that are located within a single hop distance, i.e, maximum distance allowed for transmission between two stations. These messages can be generated by all ITS stations as long as they participate in a V2X network. The receiving CAM messages provide a awareness of other stations locations, movement, basic attributes and basic sensor information.

The CAM messages are periodically generated and are used in the following cases:

Use case	**min Frequency (Hz)**
Emergency Vehicle Warning	10
Slow Vehicle Indication	2
Intersection Collision Warning	10
Motorcycle Approaching Indication	2
Collision Risk Warning	10
Speed Limits Notification	1 to 10
Traffic Light Optimal Speed Advisory	2

Table 2.2: CAM Use cases *Source: [11]*

At the moment CAMs are used in seven cases as table 2.2 presents. The generation and minimum transmission rate is 0,1 seconds. This messages also require a minimum latency of 100 milliseconds.

Decentralized Environmental Notification Message (DENM)

A Basis Set of Applications (BSA) is defined by the ETSI TS 102 637-3 [12]. The BSA contains a RHW application that is distributed among OBUs and RSUs. This application broadcasts useful information about road conditions. A DENM is triggered by RHW events, to provide information about a specific driving event or environment, experienced by a vehicle, to other vehicles. The DENM receiving ITS station should be able to provide appropriate HMI information to the driver, who makes use of this information to take action in his driving.

According to [12] DENMs are transmitted when the RHW events presented by table 2.3 occur. These messages are not periodical as the CAMs are. DENMs are triggered by events that can impact road safety or traffic efficiency. For efficiency purposes these messages can be disseminated over a long distance to allow traffic management. The events in general are characterized by a type, location, detection time and duration. DENMs frame formats include a header, a management field, a situation field and a location field. The message format is specified in [12] and contains the following information:

- ITS Protocol Data Unit (PDU) header
  - Protocol Version - Indicates the current version of the protocol being used at the management container level.
  - Message ID - Message type identifier associated to DENM
  - Generation time - Time stamp when the DENM is generated in Coordinated Universal Time (UTC).
- Management
  - Originator ID - ITS station identifier
  - Sequence Number - Sequence number provided by the originator when an event is detected for the first time.
  - Data version - Data version indicating an update of the event evolution.
  - Expiration time - In UTC
  - Frequency - Transmission frequency of DENM as defined by the originator ITS station.
  - Reliability - Probability for the event information to be true.
  - IsNegation - Negates the existence of the event
- Situation
  - CauseCode - Identifier of the event direct cause.
  - SubCauseCode
  - Severity - Severity value of the event.
- Location container
  - Latitude - Latitude of the event reference position.
  - Longitude - Longitude of the event reference position.
  - Altitude - Altitude of the event reference position.
  - Accuracy - Event position accuracy

Use case	Triggering condition	Termination condition
Emergency electronic brake light	Hard breaking of a vehicle	Automatic after the expiry time
Wrong way driving warning	Detection of a wrong way driving by the vehicle being in wrong driving direction	Vehicle being in the wrong way has left the road section
Stationary vehicle accident	e-Call triggering	Vehicle involved in the accident is removed from the road
Stationary vehicle - vehicle problem	Detection of a vehicle breakdown or stationary vehicle with activated warnings	Vehicle is removed from or has left the road
Traffic condition warning	Traffic jam detection	End of traffic jam
Signal violation warning	Detection of a vehicle being violating a signal	Signal violation corrected by the vehicle
Road-work warning	Signalled by a fix or moving roadside ITS station	End of the roadwork
Collision risk warning	Detection of a turning/crossing/merging collision risk by a roadside ITS	Elimination of the collision risk station
Hazardous location	Detection of a hazardous location	Automatic after the expiry time
Precipitation	Detection of a heavy rain or snow by a vehicle (activation of the windscreen wrappers)	Detection of the end of the heavy rain or snow situation
Road adhesion	Detection of a slippery road condition (Electronic Stability Program (ESP) activation)	Detection of the end of the slippery road condition
Visibility	Detection of a low visibility condition (activation of some lights or antifog)	Detection of the end of the low visibility condition
Wind	Detection of a strong wind condition (stability control of the vehicle)	Detection of the end of the strong wind condition

Table 2.3: DENM use cases, triggering and termination conditions *Source: [12]*

2.2.4 Hardware architecture

As HEADWAY is an ongoing research project based on draft WAVE and ETSI standards, there is the necessity for flexibility to quickly integrate new and upcoming specifications. To achieve this flexibility, the system was assembled to work as a prototype OBU/RSU.

The main components of the hardware architecture are the IT^2S board that it's composed by an Field-Programmable Gate Array (FPGA) module, a dual Radio Frequency (RF) frontend module and complementary electronic such as a Analog to Digital Converter (ADC) and a Digital to Analog Converter (DAC). This board also has several hardware interfaces to enable communication and integration with the vehicle and other architectural components. HEADWAY's hardware architecture is being developed to allow its use as "DSRC - Universal Serial Bus (USB) pen" on an embedded Linux computer, similar to how 3G USB pens work on general purpose computers. The hardware architecture is depicted in figure 2.4.

As mentioned previously in this section, HEADWAY has an implementation structure based on layers that require different resources and therefore their implementation support can differ. Because the resources requirements by the Application, Network, Facilities, Security and upper MAC layers are less intensive, these are implemented in software on an embedded Linux computer. The lower layers, PHY and lower MAC, perform more time critical operations such as synchronization, channel coordination and bit oriented operations, therefore are more efficiently implemented in hardware.

The PHY and lower MAC layers are implemented in hardware. The digital PHY and lower MAC are implemented in a FPGA for flexibility purposes, and, the analog PHY uses ADCs, DACs and a dual RF frontend module for transceiver, amplifier and RF switching implementation. The software and hardware implemented layers are connected through USB for data exchange and hardware upgrades, allowed by the FPGA. For wireless communication between vehicles, an antenna connector is used to plug the DSRC 5.9 GHz antenna. The integration with the vehicle is throughout an On-Board Diagnostic version 2 (OBD-II) interface that is used to access vehicle sensor informations such as speed, status error codes, etc.

At the moment the embedded Linux computer at use is a Raspberry Pi. This micro processor allows USB and Bluetooth connectivity.

Figure 2.4 also depicts that the HEADWAY already expects smartphone integration to provide a GUI. The smartphone should be connected with the embedded Linux computer through USB or Bluetooth connectivity.

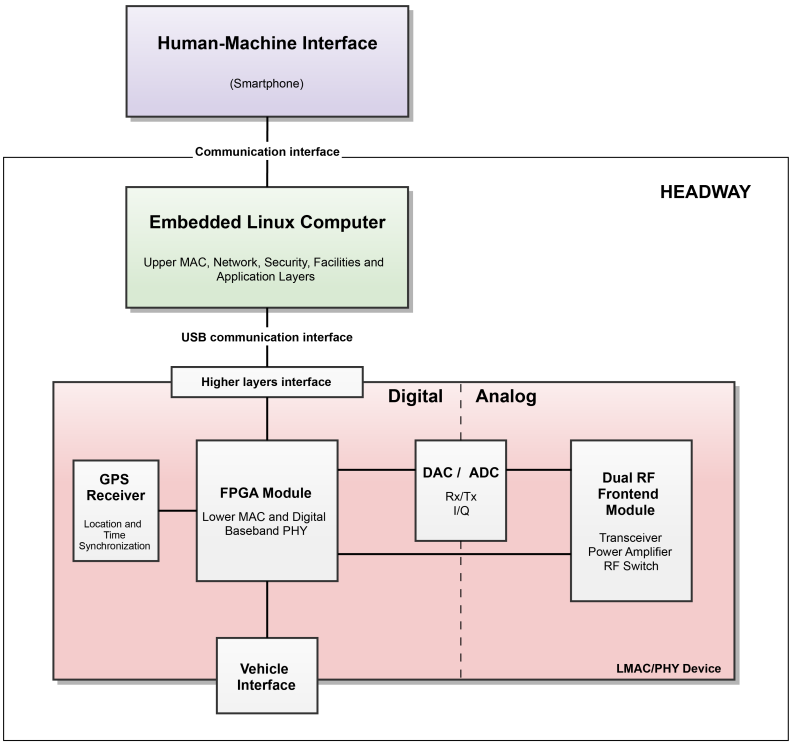


Figure 2.4: HEADWAY hardware architecture. *Adapted from [10]*

2.3 Smartphones

Since the launch of the iPhone, back in 2007, the smartphone industry has been constantly innovating and releasing new products to the market. Smartphones today are very powerful devices, with several hardware features packed in a small and portable handset.

Smartphones today, pack a diverse set of technologies. Generally these devices possess powerful processors, high resolution displays, sensors such as accelerometers, magnetometers, gyroscopes, location systems and high resolution cameras. They also pack several connectivity options such as NFC, Bluetooth, Wi-Fi, LTE/UMTS/GSM and USB. Adding to this, these devices run a mobile oriented Operating System (OS), that allow the development of applications that can take advantage of these features in a mobile context.

These mobile OSs, usually, allow for any person to develop applications. Many companies, in diverse areas, are bringing their services to these devices. Banks, e-commerce platforms, games, social networks, book editors, news media and transport are some examples of that. This is an indicator of the smartphone potential.

2.3.1 Relevant mobile platforms

The smartphone popularity is increasing. According to the market intelligence agency Internet Data Corporation (IDC) [13], in 2010, 305 million smartphone devices were sold globally and projections point towards 1161 million sold devices in 2016. Figure 2.5 presents the smartphone shipments per year predictions from 2010 to 2016.

This is a profitable market and several manufacturers like Apple, Samsung, LG, Sony, HTC, Huawei, ZTE among others, have their products available for consumers. The same manufacturer usually provide a wide range of devices with different characteristics, which gives the consumer a wide range of choices.

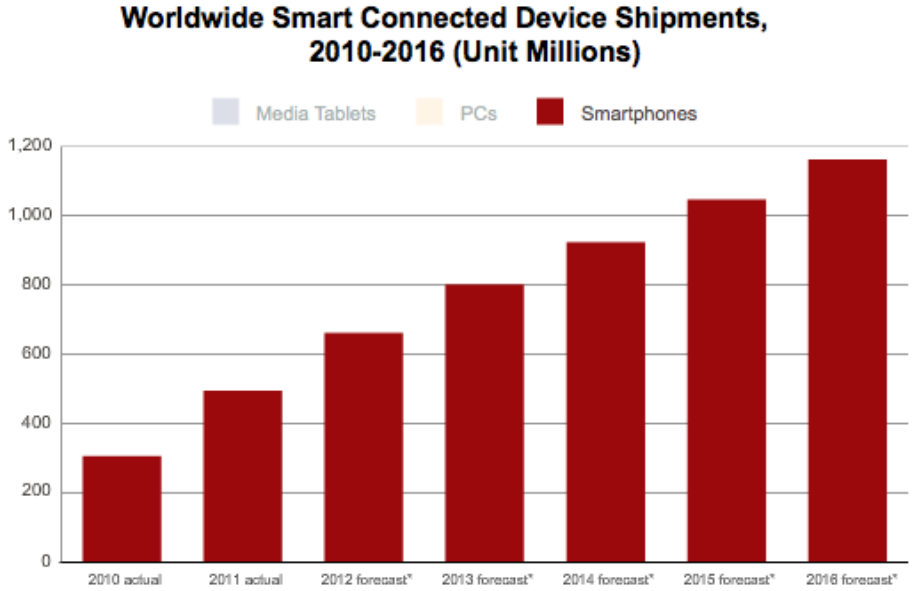


Figure 2.5: Worldwide smartphone shipments per year predictions (Millions of Units). *Source: [13]*

The predictions presented in figure 2.5 are from 2011, but they still provide a valid estimate of how fast the smartphone market is growing. This growth means that the number of people carrying a smartphone is increasing every year.

Despite the number of different devices in the market, these can be grouped by platform. Mobile platforms include the OS and the tools to develop, design and distribute applications. An application can be installed and executed on smartphones from different manufacturers, if the OS that it runs, is the one targeted by the developer. From the developer point of view its important to analyse the market by mobile operating systems and not by device. Table 2.4 presents Gartner market analysis for the first quarter of 2012 and 2013. This analysis only provides a snippet over the market share distribution over platforms, but is still a valid insight over the market progression.

Operating System	**1Q13 Units**	**1Q13 Market Share (%)**	**1Q12 Units**	**1Q12 Market Share (%)**
Android	156,186.0	74.4	83,684.4	56.9
iOS	38,331.8	18.2	33,120.5	22.5
BlackBerry	6,218.6	3.0	9,939.3	6.8
Microsoft	5,989.2	2.9	2,722.5	1.9
Bada	1,370.8	0.7	3,843.7	2.6
Symbian	1,349.4	0.6	12,466.9	8.5
Others	600.3	0.3	1,242.9	0.8
Total	210,046.1	100.0	147,020.2	100

Table 2.4: Worldwide smartphone sales to end users by operating system in 1Q13 (Thousands of Units), May 2013. *Adapted from: [14]*

Even if the number of manufacturers and different devices is significant, there are only two platforms that clearly dominate the market, those are Android and iOS. From the developer perspective, when considering developing an application, iOS and Android should be the platforms targeted.

In subsections 2.3.2 and 2.3.3 is described two different methods to develop applications, the platform resources and the tools available.

2.3.2 Native development

Native application development is the concept of using native programming languages, frameworks and Application Programming Interface (API) to develop an application. Most relevant mobile platform provide developers the necessary tools for this type of development.

When developing with this method the resulting application can be as efficient and stable as possible. Different mobile operating systems work in different ways, therefore before starting the development of applications for a specific OS a learning process must take place. Applications targeted for one OS will not run on another.

Android and iOS are the most dominant operating systems today. Because of this dominance there is a great interest by the development community in developing applications for this platform, therefore community support is rich.

The main features and characteristics of these two mobile operating systems are described in the following subsections.

Android

Android is a platform that can be used both for mobile devices like smartphones and tablets as well as for embedded systems. Android results of an open-source project named Android Open Source Project (AOSP) from the Open Handset Alliance (OHA) led by Google [15]. The AOSP is responsible for developing and maintaining Android. Since 2008 when the first device running Android was released to the public, this platform has been quickly evolving and at the moment this was written the most recent version is 4.2.2 [16].

Android provides tools and services for developers to use in application design, development and distribution.

In order to set a standard for the User Experience (UX) across different applications, Android developers website [17] provides guidelines to application design.

For development, Android provides a Software Development Kit (SDK) that can be integrated in Eclipse Integrated Development Environment (IDE) through Android Development Tools (ADT) plugin. All the necessary tools and information for application development can be found in Android developers website [17].

For distribution, Android has Google Play application repository. To be able to publish an application in this repository developers have to pay a 25 USD per year fee and 30% of application revenue is collected by Google [18].

Through the SDK, developers have access to a wide range of frameworks and APIs. According to [19] the most noteworthy SDK features are the following:

- Access to hardware, including camera, GPS and sensors
- Data transfers using Wi-Fi, Bluetooth and NFC
- Maps, geocoding and location-based services
- Background services
- SQLite Data Base for data storage and recovery
- Content share and inter-app communication
- Media support and 2D/3D graphics
- Cloud to device messaging (C2DM)
- Optimized memory and process management

The main programming language for Android application development is Java. It also supports C and C++ through is Native Development Kit (NDK) also available in the developers website. This is used to implement parts of the application that require higher performance. Each application in Android runs in its own process and instance of Dalvik Virtual Machine (VM), therefore allowing for multitasking. This VM depends on Linux kernel for memory management and threading.

iOS

iOS is an proprietary OS from Apple for its mobile devices iPhone, iPad and iPod. This OS is based on Mac OS X, sharing the same kernel, Berkeley Software Distribution (BSD) [20] sockets for networking and Objective-C and C/C++ compilers [21]. iOS was first launched to the market in 2007 with the first Apple iPhone.

There are two necessary requirements to develop native applications for iOS. The first is to own an Intel-based Mac running Snow Leopard or later and the second is to be registered as an Apple Developer and enrol in iOS developer program [22]. After these two requirements are fulfilled developers have access to iOS SDK that should be used with the Xcode IDE.

For distribution, iOS has App Store application repository. To be able to publish an application in this repository developers have to pay a 99 USD per year fee and 30% of application revenue is collected by Apple [22].

Like Android, iOS also has its own design language in order to provide a standard UX across different applications. The guidelines to this design can be found in iOS developers website [21].

Through the SDK the developer has access to the Cocoa and CocoaTouch frameworks that provide several APIs. The most relevant features of iOS SDK are the following:

- Background services and multitasking
- Apple push notification service
- Local notifications
- Peer-to-peer services
- Maps and location
- Accelerometer and Gyroscope
- SQLite
- File sharing support
- Inter app messaging

The features available both in iOS and in Android are very similar. Both offer a complete set of features to develop an application.

To develop hardware accessories for Apple iOS devices that use Airplay or the proprietary cable connector to communicate, its necessary to enrol in Apple's MFi Program. For Bluetooth accessories development MFi Program its not necessary [23].

The main programming language used to develop for this OS is Objective-C and its with this language that the best performance can be achieved since the frameworks are also written with it.

2.3.3 Multi-platform development

There are tools that allow the development of applications that run in different OSs, therefore helping the developer to reach the maximum number of users with minimum effort.

In order to develop a multi-platform application there are two valid approaches. The first is to develop a Web application and the second is to develop an Hybrid application. Both concepts are explained following.

Web Application

A Web application can be developed using web programming languages like HTML5, CSS and Javascript. The application source code is interpreted by the web browser and not directly by the OS. This type of application doesn't require installation and its data is stored in a web server. To use the application the user introduces an Uniform Resource Locator (URL) in the browser.

This type of application can be updated easily due to its lack of need of installation. Once the developer updates it, every user will have access to the most recent version without any effort. One other big advantage is the cost of publishing the application being significantly lower because its not stored in any proprietary repository like Google Play or App Store from Android or iOS respectively.

With HTML5 it's possible to hide the address bar from the browser, making the application feel more native.

Web applications have limitations when accessing the device hardware resources and its important to note that its offline use its also limited.

Hybrid Application

Hybrid applications are somewhere in between of Web applications and Native applications. The programming languages used to develop these applications are the same as the Web applications, i.e, HTML5, CSS, Javascript. Using tools like PhoneGap [24] its then possible to wrap the code in a container that will be compiled to the OS native code.

These application can run offline and access all the hardware features of the device, due to the native part of the code. The performance of these applications is higher than the web applications but lower that the native applications.

2.3.4 Comparative analysis

When analysing the tools and platforms relevant for mobile application development described in sections 2.3.2 and 2.3.3 it's clear that there are two categories for mobile development. On the one hand are the Native mobile applications which are developed using the programming languages Java or Objective-C for Android and iOS respectively. On the other hand the Multi-platform tools developed using HTML5, CSS and JavaScript.

One of the characteristics of the Native applications is that they are OS specific, meaning that they can only run on the targeted platform. This method of application development as the highest time cost but results in a more stable, higher overall performance and UX.

Android OS as the advantages of being open-source and the programming language is Java, which is widely used in other development contexts and therefore more familiar. When developing natively for this platform the developer as access to all the hardware features of the device.

Like Android, developing natively for iOS means that, developers can access all the hardware features of the device. The MFi program, due to its costs, can raise problems for iOS development, if the application is intended to communicate with an USB or Airplay accessory.

Multi-platform applications main advantage is the time and cost of development and maintenance. As described in section 2.3.3 there are two different methods to develop these applications.

Web applications run in the operating system browsers, therefore are dependent on the internet connection. Although they are relatively fast to develop and require the lowest maintenance, the performance, stability and UX are in general lower when compared to native applications. Another disadvantage is the limitation in accessing the device hardware features.

Hybrid applications try to bring together the best in both worlds. They combine the advantages of Web applications with some of the Native applications stability, performance and user experience.

In table 2.5 its summarized the highlights of this analysis.

	iOS	**Android**	**Web**	**Hybrid**
Development Skills	Java	Objective-C	Javascript, HTML, CSS	Javascript, HTML, CSS
Maintenance Cost	Highest	Highest	Lowest	Medium
Connectivity	Online and Offline	Online and Offline	Mostly Online	Online and Offline
Hardware access	Total (Native frameworks)	Total (Native frameworks)	Limited	Total
Performance	Highest	Highest	Lowest	Medium
User experience	Native	Native	Emulated	Emulated

Table 2.5: Mobile application development methods comparison

The decision of which method to use in mobile application development should be based on the type and purpose of the application itself.

2.4 Related projects and applications

Smartphones are powerful devices. With their hardware features, API's and frameworks, its possible to develop applications in several different areas. Many applications have already been developed in order to introduce some level of intelligence into transportation systems. Instead on focusing only in application that implement an AAD and the eCall, this search ranges ITS areas such as safety, comfort, information and overall efficiency categories.

A search was made in order to assess what is already available to drivers. This search followed two different approaches. Application repositories from the most relevant mobile operating systems where searched in order to assess what is already there for the driver to use. In an attempt to understand what the scientific community is developing in this area, a search through the most relevant article and reports repositories was also conducted. In sections 2.4.1 and 2.4.2 the result of this search are presented.

2.4.1 Applications available in platform repositories

In both Android and iOS application repositories its possible to find a very large number of applications for every category. The goal of this search was to assess if there were applications available that provided services in the three categories described in 2.1.1, namely, safety increase, environment preservation and transportation efficiency. Both in Google Play and in App Store there are many applications that have no quality assurances or that replicate services. The result of this search presents only some examples of applications with high rating and user feedback, is mentioned in this subsection. The relevant findings are presented following:

- Avertino - A safety increase application. Generates permanent, regular or even temporary road danger location based warnings. These events are reported by application users and are subject to confirmation by other users. When approaching a marked location the application warns the user through a visual and audible alert. It also provides the possibility to visualise the reported event on a map. Available for iOS and Android.

- CoPilot Live - Efficiency increase application. Provides turn by turn navigation applications with life traffic information. When traffic conditions change in the drivers route, the application suggests an alternative faster route. Available for iOS and Android.

- Waze - Safety and efficiency increase application. Provides turn by turn navigation application that allows the users to report several events such as road dangers, accidents, traffic jams, speed cameras and police vehicle inspections. It works based on the user community reports. It also suggests faster route alternatives to drivers based on traffic alerts. Available for iOS and Android.

- ParkDroid - Efficiency increase application. Helps to locate an available parking spot based on the user current location. The application shows a map with the location of different parking lots and indicates if there are available spots through the map pin color. It allows the user to navigate directly to the parking lot with turn by turn directions. When parked, the user as the possibility to save the car's location and later ask turn by turn walking directions to it. Available for Android.

- eco:Drive - Efficiency increase and environment preservation application. Works with FIAT vehicles equipped with the Blue&Me technology. It gathers travel informations such as fuel consumption, travelled distance, CO2 emissions and spent fuel cost. This information is displayed to the user to help and motivate him to perform better and reduce his ecological footprint. Available for iOS and Android.

- iOnRoad Augmented Driving - Safety increase application. Uses augmented reality to analyse in real time, using the device camera, all the objects that are in front of the vehicle while the user drives. It generates alerts when the user is not respecting the minimum safety distance between cars or when an exit/entrance is approaching. Available for iOS and Android.

At the time of this search no application was found that implemented an AAD and an eCall help request.

2.4.2 Applications as a result of research projects

Other than the applications that are already available in applications repositories, projects and concepts are also being developed by the scientific community that try to explore the smartphone capabilities in transportation systems. The result of this search presents some examples of relevant projects that are related to this dissertation. The results are the following:

- Girts Strazdins et al. [25] describes a project developed to use smartphone to assess road surface quality. The application takes advantage of the hardware features of modern smartphones like accelerometers and GPS. The sudden movements that occur when a car is upon a pothole produces acceleration values that can be interpreted in different ways. These accelerations along with their location are reported through internet to a web service to later be delivered to other application users. The potholes are displayed on a map that is displayed to the users in the application.

- Jorge Zaldivar et al. [26] describe an Android application that connects to the car OBD-II system through Bluetooth. The application goal is to detect accidents using smartphone accelerometer and OBD-II airbag signal. When a 5g ($g = 9.8\ \frac{m}{s^2}$) or higher acceleration is detected or when the airbag signal is triggered the application starts a one minute countdown timer. If the user does not cancel the countdown the application will validate the event as an accident and will start the automatic help request procedure. This procedure consists in sending and email and an Short Message Service (SMS), with informations regarding the accident such as location, to a specific service and to predefined user contacts. After the SMS and email are sent a call is performed to the emergency services.

- Andrew Austin [27] describes the development of a Android smartphone application with the goal of helping its users driving more ecologically. The application uses the device GPS to determinate the vehicle mean speed and location and the accelerometer to detect harsh breaking or harsh accelerations. These informations are used to calculate the vehicle CO_2 emissions, through means of an algorithm developed for the effect. The application displays real-time information in a graphic to motivate users to perform better next time. The application also displays the location where the user performed badly to provide awareness and help to improved ecological driving.

- Drive In [28] is an under development vehicular communications project, which the IT-Aveiro is associated. This system aims to provide V2V communications to help with road traffic management and road safety. The system allows for a car that is being surpassed to be aware of the situation and thus prevent lane change. Another feature is the possibility to stream video between vehicles. This way a driver that is behind a large vehicle and has weak visibility to what's ahead can watch a streamed video recorded by that same large vehicle and be able see through it. This project is currently being tested in Porto in a taxi firm with 500 vehicles.

The scientific community is aware of the smartphone capabilities in ITS and AADs are already developed using smartphones. These are some of the projects found that are relevant. Some search results describe similar concepts and therefore are not mentioned. In all search results that involved a smartphone, the platform used was Android. Despite smartphone

based AADs already exist, no project was found that integrates this mechanism with vehicular communications.

2.5 Accident detection

An accident can occur in different ways. Car accidents happen mostly due to collisions and rollovers. Collisions can take place in multiple directions. They can be frontal, lateral, rear or even diagonal. Any of these collision directions are possible when driving a car. To develop an effective AAD mechanism both these types of accidents should be detected.

2.5.1 Collision detection

A collision generates a sudden change of speed and it happens when an object impacts another object resulting in a change of speed. The severity of the collision will depend on the direction, orientation and speed of the objects colliding. If the objects are moving in the same direction but with different orientation, the collision will be more violent than if both objects are moving in the same direction and orientation. This means that the bigger the relative speed is between the objects, the bigger the violence of the collision. A change of speed over time is called, in physics, acceleration.

The acceleration generated during an accident is a subject studied by the scientific community. Authors such as Weiner in [29], Thompson et al. in [30] and Kumar et al. in [31], describe in their publications accident detection systems, that use the 4g (g = 9.8 $\frac{m}{s^2}$) threshold, about which an accident takes place. Thompson et al. also show that smartphone falls and harsh car breaks are unlikely to surpass the 4g threshold, which proves that this threshold acts as a correct filter for false detections.

European road restraint systems are used to reduce the severity of accidents of vehicles leaving the road. To achieve this, these systems are evaluated based on the European standard EN 1317 [32] [33]. This standard is based on the Accident Severity Index (ASI). Table 2.6 presents the ASI scale values.

Impact severity level	**Index values**
A	ASI $\leq$ 1.0 and THIV $\leq 33km/h$
B	ASI $\leq$ 1.4 and THIV $\leq 33km/h$
C	ASI $\leq$ 1.9 and THIV $\leq 33km/h$

Table 2.6: Accident Severity Index (ASI) and Theoretical Head Impact Velocity (THIV) scale values. *Adapted from: [33].*

This scale measures a collision impact severity and is divided in three levels. Impact severity level A is the less severe and C is the most severe. Level A designates light injury if any. This means on level B and above there is the risk of serious injury.

Studies performed by Gabauer et al. in [34] and Shojaati in [35] demonstrate the relation between ASI and both Head Injury Criteria (HIC) and Abbreviation Injury Scale (AIS). Both HIC and AIS are metrics used to describe the injury severity of a vehicle occupant.

The Theoretical Head Impact Velocity (THIV) its not relevant to acceleration based accident detectors .

ASI determination is possible using acceleration measurements. EN 1317-1 [32] describes how to calculate ASI values.

To determine ASI its required a tri-axial accelerometer to measure longitudinal (A_x), lateral (A_y) and vertical (A_z) acceleration components. Figure 2.7 depicts the components direction and orientation. The procedure to compute ASI is the following:

1. Record the acceleration components (A_x, A_y and A_z) values.

2. Filter data with a four-pole phaseless Butterworth digital filter

   (a) Evaluation components

   $T = \frac{1}{S}$ = sampling time in seconds, S = sample frequency.

   $CFR = 13Hz$ = filter cut-off frequency

   $W_d = 2\pi CFR$

   $W_a = tan(W_d \frac{T}{2})$

   $a_0 = \frac{{W_a}^2}{1+\sqrt{2}W_a+{W_a}^2}$

   $a_1 = 2a_0$

   $a_2 = a_0$

   $b_1 = \frac{-2({W_a}^2-1)}{1+\sqrt{2}W_a+{W_a}^2}$

   $b_2 = \frac{-1+\sqrt{2}W_a-{W_a}^2}{1+\sqrt{2}W_a+{W_a}^2}$

   (b) For each of the three acceleration components, if $X(k)$ is the k^{th} element of any series of measurements and $Y(k)$ is the k^{th} element of the filtered series, then,

   $$Y(k) = a_0X(k) + a_1X(k-1) + a_2X(k-2) + b_1Y(k-1) + b_2Y(k-2) \quad (2.1)$$

   Equation 2.1 is a two pole filter. To perform the required four-pole filter data should pass to the filter twice (Equation 2.2). Figure 2.6 depicts how the data is filtered.

   $$Z(k) = a_0Y(k) + a_1Y(k-1) + a_2Y(k-2) + b_1Z(k-1) + b_2Z(k-2) \quad (2.2)$$

   (c) Compute ASI as a function of time:

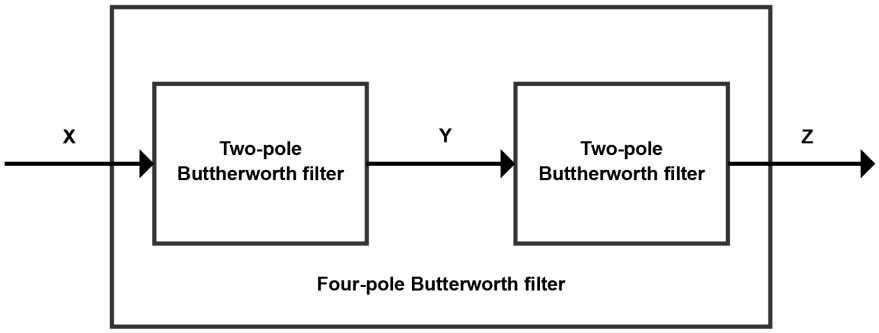


Figure 2.6: Butterworth four-pole filter.

$$ASI = \sqrt{\left(\frac{\bar{A}_x}{\hat{A}_x}\right)^2 + \left(\frac{\bar{A}_y}{\hat{A}_y}\right)^2 + \left(\frac{\bar{A}_z}{\hat{A}_z}\right)^2} \tag{2.3}$$

Where $\bar{A}_x$, $\bar{A}_y$ and $\bar{A}_z$ are the filtered components of vehicle acceleration and $\hat{A}_x$, $\hat{A}_y$ and $\hat{A}_z$ are threshold values defined in EN 1317 [32]. For vehicle occupants with the seatbelt fasten, $\hat{A}_x = 12$, $\hat{A}_y = 9$ and $\hat{A}_z = 10$.

(d) ASI should be calculated to at least two decimal places and report to one decimal place by mathematical rounding, i.e, $1,44 = 1,4$ and $1.45 = 1,5$.

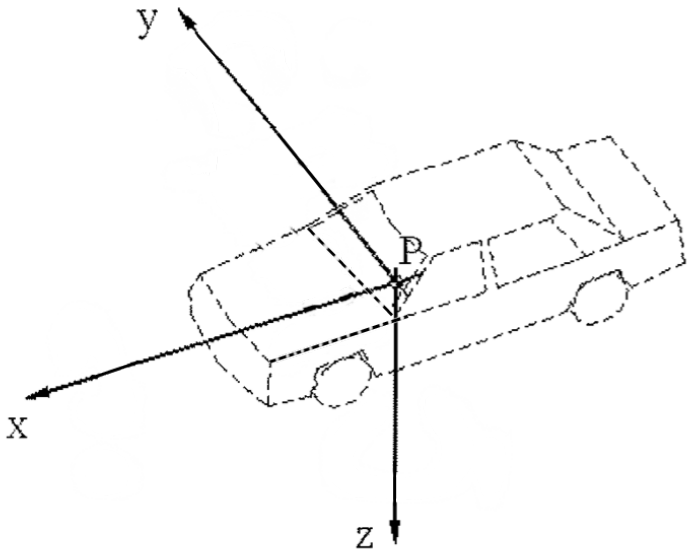


Figure 2.7: Accelerometer axis direction and orientation. *Adapted from [32].*

2.5.2 Rollover detection

Other than collisions, car accidents can be caused by rollovers. A roll over can happen in different ways and its difficult to detect using acceleration. A rollover can be easily detected if the position of the car relative to the ground is known at any moment.

Considering that in figure 2.71 the axis rotate with the car, i.e, they are fixed to the car, a rollover occurs when the vehicle rotates at least 45° over the X axis.

2.6 eCall

In 2012 there were 28000 fatalities in EU roads and this was the lowest number since 2001 [1]. The time to an injured person receive proper care from the Emergency Medical Systems (EMS) is related to the probability of death and trauma. According to Henriksson et al. [36] death and trauma rates can be reduced if there is a quicker reaction from the EMS. A quicker reaction can be obtained if help is requested immediately after the accident event occur. Also if the exact location of the accident along with other extra information is provided to the EMS, a quicker and better response is possible.

The European Commission in an attempt to provide a faster response from European EMS, declared the mandatory deployment of eCall in new cars after 2015 [37]. eCall is a automatic accident detector that in the presence of an accident automatically requests help to the EMS through the European 112 emergency number.

eCall chain

To perform an eCall, several technological aspects must be implemented in all the intervening parts of eCall chain. This chain is depicted in figure 2.8.

There are three parts involved in the eCall chain. These are the car manufacturers, the Mobile Network Operator (MNO) and the participant countries. The European Commission adopted regulatory measures to mandate the technology deployment and upgrade on the three parts. Each part is responsible for upgrading their involved technology according to the eCall specifications.

As figure 2.8 depicts when a accident occurs, the car system performs an eCall that is composed by the voice call and a Minimum Set of Data (MSD) that is also transmitted, through the MNO, to the most appropriate Public Safety Answering Point (PSAP).

The solution adopted for the MSD transmition, is an in-band modem that transmits data in the voice channel [39]. This modem along with the accident detection system will be deployed in new cars after 2015.

The MNO, in order to support this type of connection, need to upgrade their networks before 2015. European Commission also mandates the upgrade Member Countries PSAPs so that the information received, as a MSD, can be properly analysed and the EMS dispatched quickly.

The MSD should contain information to help speed up the EMS arrival to the accident scene. According with the eCall Driving Group recommendations [40] the MSD should be sent in a 140 Bytes containing the following information:

- Control - One Byte to specify if the eCall was automatically or manually triggered, if its a test call and if there is confidence in the location provided.

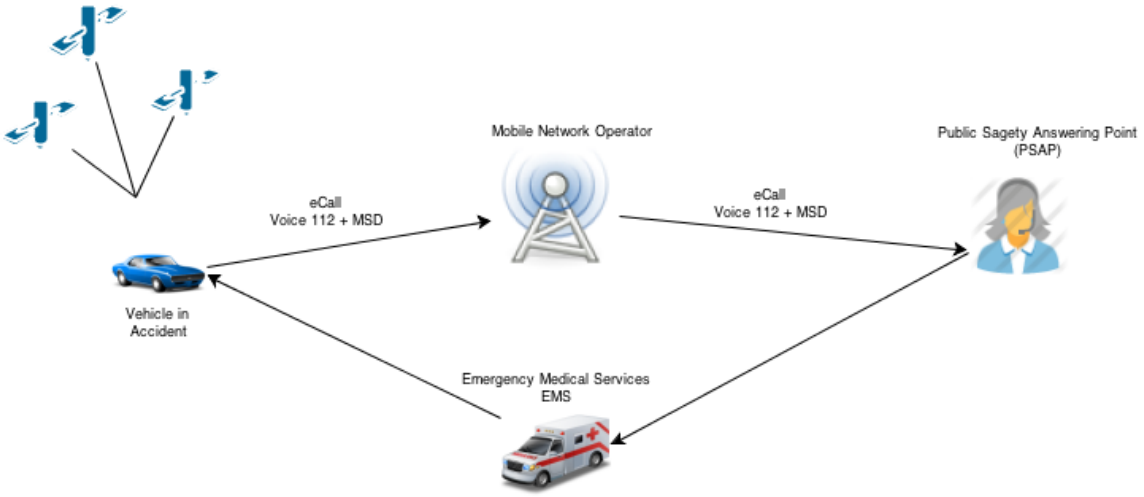


Figure 2.8: eCall chain. *Adapted from [38]*

- Vehicle Identification Number (VIN) - Twenty Bytes for sending VIN according to ISO 3779 norm. Because VIN database for national and foreign registered vehicles might not be available in all Member States, the advantages of using this information on the PSAP should be further assessed.
- Timestamp - Four Bytes. The time of the accident should be provided in the UTC metric.
- Location - Latitude (four Bytes), Longitude (four Bytes) and direction of travel (one Byte) based on the last three positions.
- Service Provider - Four Bytes for service provider Internet Protocol (IP) address in Internet Protocol version 4 (IPv4). Optional field.
- Optional data - Up to 106 bytes for other informations. Optional field.

The eCall system is supposed to work seamlessly in all participant countries, i.e, if a driver as an accident with his vehicle, in a foreign country, the eCall is performed to one of the PSAPs of that foreign country.

In the eCall Driver Group recommendations [40], the eCall service chain can be found. There are six main domains of this chain as it depicted in figure 2.9. Each domain description is presented following:

- Vehicle eCall Triggering System - Compose by sensors that should detect front, side, rear and roll crashes. The trigger should be generate by the airbag module and/or a combination of other sensor data (e.g. gyro, radar, speed. Trigger thresholds based on speed variations could also be sent as optional data to help PSAPs predict the likeliness of serious injuries. The eCall can also be triggered manually.
- eCall Generator (EG) - In vehicle software triggers the eCall, provided the necessary info from the triggering system, and initiates the 112 call and MSD transmission through the in-band module.

- EG to MNO - The network receives the 112 call and the MSD.
- MNO - The mobile network operator (MNO) enriches the 112 call with Caller Line Identification (CLI), MSD and cellular location.
- MNO to PSAP - Forwards the enriched 112 call to the appropriate PSAP.
- PSAP - Answers 112 voice call, decodes and visualises cell location and PSAP.

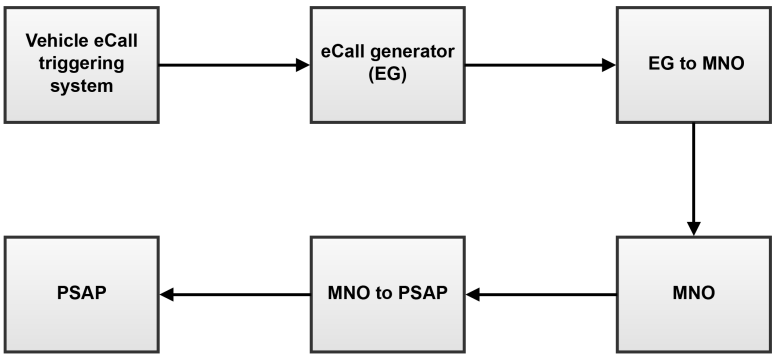


Figure 2.9: eCall service chain domains. *Adapted from [40]*

From this chapter is important to retain what HEADWAY is about and how its being developed. Its also important to understand what the eCall project is about and its purpose. The accident detection is one of the main goals of this dissertation. This chapter assessed how accidents could be detected using smartphone sensors.

Having the background knowledge described, its possible to use this information to formulate a proposal. The next chapter describes this proposal and its aim is to help link the raw information gathered in this chapter to the actual application development.

Chapter 3

Proposal

As mentioned before in section 2.1.2, vehicular networks are composed mainly by OBUs and RSUs. In the in-vehicle domain of these networks, an AU should coexist along with an OBU. This AU should deploy applications that take advantage of the OBUs communication capabilities in order to provide services in safety, efficiency and other categories. The AU should also provide a User Interface (UI) to allow drivers to receive or transmit messages such as RHWs throughout the network. This unit can be either integrated in the same device as the OBU, or it can be a physically separated device.

In section 2.6 it's presented the eCall system that was made mandatory, by European Commission, for all new vehicles starting 2015. This system predicts an AAD mechanism that in case of accident detection, automatically requests help to the EMS through a voice call and the MSD.

In section 2.2 HEADWAY is described. This vehicular communication solution is integrated in the VANET concept as an OBU and RSU. This system implements ETSI and IEEE standards and an AU is required to deploy applications that can demonstrate the system to potential investors.

The goal of this dissertation is on the one hand, to develop a smartphone based AAD and help request, which is basically the automobile part of the eCall system, that could be integrated in vehicles produced before 2015 and, on the other hand, to use the smartphone to demonstrate HEADWAY's potential. The proposal aims to line up HEADWAY's requirements and the eCall system requirements in order to develop a smartphone phone based application that could work as a solution for both requirements and still integrate them.

This chapter describes the proposal. First the smartphone use as an AU is described. HDy Copilot's eCall system along with other features to implement are also proposed. Finally the proposed implementation and integration are described.

3.1 Smartphone as an Application Unit (AU)

At this early phase the HEADWAY doesn't contain an AU. This unit should be used to develop applications to take advantage of V2X communications. Smartphones due to their hardware resources and software capabilities allow the development of applications that can do just that.

As mentioned in section 2.2, HEADWAY implements the Facilities layer from the ETSI protocol stack. This layer generates a well defined type of messages (CAMs and DENMs) to

be transmitted in V2X communications. These messages are the core of V2X functionalities. An AU should be capable of using the received messages and generate them, namely DENMs, due to its nature.

The DENMs are triggered by particular events named RHWs. These events are road safety and efficiency related. There are several predicted RHW use cases that are carried by DENMs as table 2.3 presents. Some use cases can be easily detected by the driver, such as traffic condition, hazardous locations, stationary vehicle - vehicle problem. Others such as stationary vehicle accident should be triggered by an eCall system. HDy Copilot should provide the resources to report these exemplified RHW use cases.

Smartphones pack several hardware resources that can put to use to develop an AU. The large screen in smartphones allow them to display a GUI, that can be used has HMI for HEADWAY, allowing RHW presentation/report to/by the driver. The rich sensor resources, such as accelerometer and gyroscopes, can be used to develop the AAD mechanism required for the eCall system. With the smartphone connectivity implement the GSM connectivity the eCall help request could be implemented.

Using an smartphone based AU an application prototype could be developed to implement these features and therefore line up HEADWAY's requirements and the eCall system requirements.

3.2 Application main guidelines

This dissertation is attached to the ITS context. Its main goal is to develop a smartphone based eCall system (AAD mechanism with help request) and integrate the smartphone with HEADWAY. To achieve this, a proposal was presented to use the smartphone has an AU and line up all the requirements to achieve the goals.

Using a smartphone as an AU, allows the use of its rich hardware resources to deploy applications in safety and efficiency and to develop an HMI, that comply with the goals intended for HEADWAY, described in section 2.2.

From chapter 2, with the description of the background knowledge of the system functionalities, is possible now to create guidelines to be followed in order to develop a valid AU.

The guidelines presented in this section, aim to provide a link from the background knowledge to the actual solution proposal. To achieve the dissertation goal, HDy Copilot should provide the means to receive and transmit RHWs from/to other vehicles connected to the network. To demonstrate HEADWAY some RHWs should be detected manually by the user, in order to be possible to exchange a DENM on-demand. The rich hardware resources, present in smartphones, such as sensors and connectivity should be put to use to develop an eCall system that would detect accidents, request help and generate a RHW to be transmitted through HEADWAY. To accomplish these goals, two features are purposed following:

- GUI - User interface to allow direct interaction with HEADWAY, namely, incoming RHW info display and mean for the driver to manually report RHWs.
- eCall system - Means to detect accidents automatically (AAD), request help like eCall (call and data) and generate a RHW to be transmitted to the OBU (HEADWAY).

This proposal refers to a smartphone based prototype application and its guidelines are more thoroughly explained along this section.

3.2.1 GUI

The GUI is an important part of the application because it acts as a HMI. HEADWAY is supposed to be used in a vehicle and therefore in a mobile environment. The GUI is supposed to be used during a mobile environment, namely, the driving act. This raises safety concerns. It's design and UX should take into account the safety of the vehicle occupants.

The GUI should provide direct access to the services deployed in the system, whether they are safety, environment preservation or efficiency related. To provide a solid and contextualized UX, the interaction should mimic the retail navigation systems commonly used. The displayed content must allow a quick interpretation of data, to prevent the loss of focus on driving. The interaction with services must then be designed to require minimal interaction steps, so that the user stays focused on the road. Also due to its context of use, the interface should be responsive and reliable, so it can provide confidence to the driver and add value to the system.

The AU should be placed in the vehicle similarly to navigation systems. The use of a smartphone car holder is a suitable solution. This way the driver can place the smartphone car holder in a position of his choice. Once the AU is placed the driver should be capable of glancing the device to interpret data as well as easily reach the device for interaction.

The interaction with the system should consist mainly in the display of incoming RHW and the means to manually report them. As this system is a proof of concept only three RHW use cases will be implemented for manual detection. These are use cases that can be easily detected by the driver. The use cases are:

- Traffic condition
- Hazardous location
- Stationary vehicle - vehicle problem

eCall system

The eCall system is composed by two parts, the AAD and the help request. Smartphone based automatic accident detection is not a new concept as described in section 2.4.2. This dissertation however, has the particularity of proposing an integration with a VANET, which would allow the AAD to trigger a DENM.

To implement an AAD an Accident Detection Algorithm (ADA) is proposed. The algorithm should detects car accidents by using the device hardware resources along with the vehicle OBD-II signals, which are available through the HEADWAY. As a response to an accident detection the smartphone connectivity should be used to perform an eCall and a generate a RHW to be transmitted to the OBU in order to trigger a DENM to warn other drivers.

3.3 Proposed implementation

This section describes the proposal of the use of a smartphone as an AU. The AU should display a GUI that allows interaction, namely, RHW display and report. The system should also detect accidents and perform an eCall help request. The AU should allow its use without compromising the safety of the vehicle occupants. This section proposes the functionalities

to be implemented in the three major parts of HDy Copilot, that are, the ADA the RHW manual report and the GUI.

3.3.1 Accident Detection Algorithm (ADA)

The eCall system predicts the existence of an AAD. The rich hardware resources of the smartphone allow, as presented in section 2.4 the development of an AAD mechanism. This mechanism should be based on a ADA that uses gathered information to validate accidents and proceed with the appropriate responses.

To detect an accident is important to perceive the vehicle behaviour. Smartphones pack accelerometers, gyroscopes, magnetometers and location systems. If the smartphone is attached to a car holder, the vehicle and the device possess an inelastic bound. This means that the forces applied to the vehicle are the same as the smartphone. This allow the smartphone to interpret the vehicle acceleration (accelerometer) at any given moment as well as its position relative to the ground (gyroscope) and speed (GPS). Also, the algorithm should be constantly monitoring the OBD-II data from the vehicle. Informations such as airbag deployment should be used in the algorithm, to validate an accident occurrence.

In case an accident occurs, the algorithm should respond by requesting help and notifying relevant entities. Its purposed that the ADA, once an accident is validated, perform an eCall, generate a RHW to be transmitted to the OBU to generate a DENM to warn other drivers and finally notify other contacts (family/friends) previously setted by the user. Figure 3.1 presents the flow chart for the algorithm.

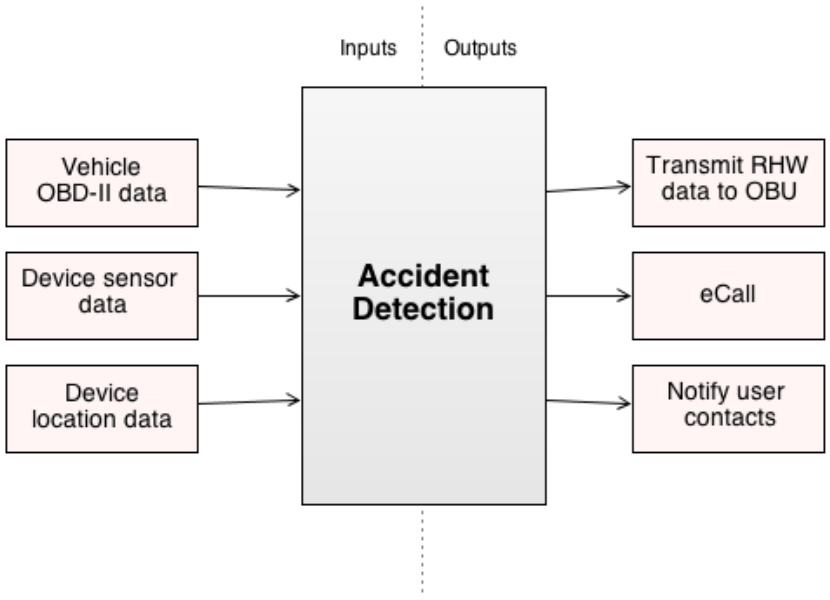


Figure 3.1: Accident detection algorithm flow chart.

The ADA should be constantly being executed, even when the application is in background, or the device is idled.

3.3.2 RHW manual report

This feature should be implemented in the GUI and provide manual access to the driver. Each RHW should have a unique identifier code. Once the user manually reports a RHW, the device's location and the time stamp are saved and this data along with the RHW identifier code should be transmitted to the OBU, so a DENM can be triggered. Figure 3.2 presents the RHW manual report flow chart. The application has three manual RHW as mentioned before. Each of these RHW should be triggered by pressing a dedicated GUI button.

Figure 3.2: Manual RHW flow chart.

3.3.3 GUI mockup

The GUI is an important part of the HDy Copilot. As safety is one of the main goals of ITS, HDy Copilot graphical design should allow a quick and easy interaction. The smartphone screen real state, should be used intelligently to easily display the necessary information to the driver, considering its position while driving. The UI elements to present should be large in size and with high contrast colors, so that they can be easily perceived, even with strong day light.

The GUI displays data to the driver and allows him to interact with HDy Copilot. Still, the user interaction with HDy Copilot should be limited, to avoid the loss of focus on the road. It is proposed that the user interact with HDy Copilot by visualising RHW that concern him, by reporting RHW to other vehicles and by visualizing vehicle sensor data. These are the main interactions that the user should have with the application. The GUI should then present three different layouts. One for the incoming messages (Received), another for the RHW report (Report) and another for the vehicle's OBD-II vehicle data (OBD-II). Each layout is described following:

- OBD-II - Displays OBD-II data. Having this data displayed can add value to HDy Copilot, since the user can check what's going on with is vehicle system.
- Received - Here is where RHWs are displayed. The driver needs to perceive the DENM content as quickly as possible, to be prepared for upcoming road events, without losing focus on driving. With this premisses DENM's data should be displayed through symbols and with minimum text content. Text can be harder to perceive without being focused.
- Report - To report RHW the user should press a graphical button, that displays the type of RHW to be reported.

To better understand this proposal a layout mockup of HDy Copilot is presented on figures 3.3, 3.4 and 3.5.

Figure 3.3 presents the Received layout. Here is where the DENM content is displayed to the driver. As described in section 2.2.3 DENM's format wrap several data. Only two fields are relevant to the driver. The type of RHW and its location. The remaining information should be used to process data in the Facilities layer.

As figure 3.3 depicts the type of RHW and the remaining distance to it is displayed as text. The large square presented in yellow color, alerts the user of an upcoming RHW down the route. Three color can be displayed in the square:

- Green - When there is nothing to report, meaning the route is safe;
- Yellow - When some RHW was reported at a fairly long distance;
- Red - When some RHW was reported at a short distance;

With this scheme the driver can easy interpret information regarding an upcoming RHW. With usage, the driver will understand the colors meaning and that is the most important. Textual information will only provide extra information.

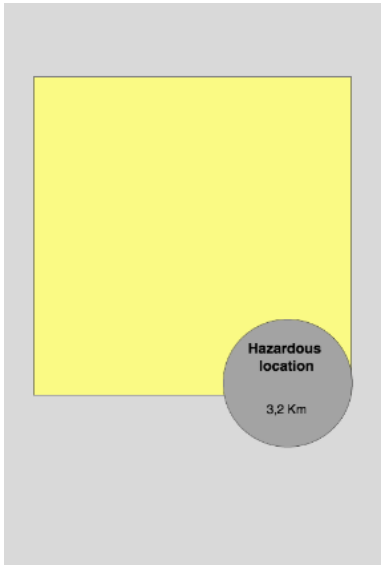


Figure 3.3: Received layout mockup.

Figure 3.4 presents the Report layout. Here the user can report any of the three RHW by taping on the button that presents the text relative to the event. The buttons should be large and present a high contrast to be easily visualized in day light and avoid missed taps. Although all layouts background color is grey in the mockup, this or any other color is final and can be changed if during development a better contrast is achieved.

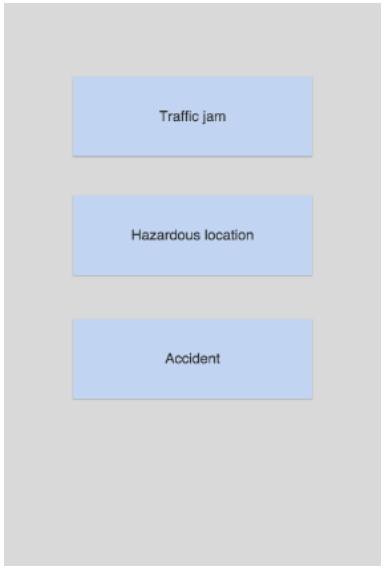


Figure 3.4: Report layout mockup.

Figure 3.5 presents the OBD-II layout. Here the user can check his vehicle sensor data. this information can be useful to detect vehicle problems.

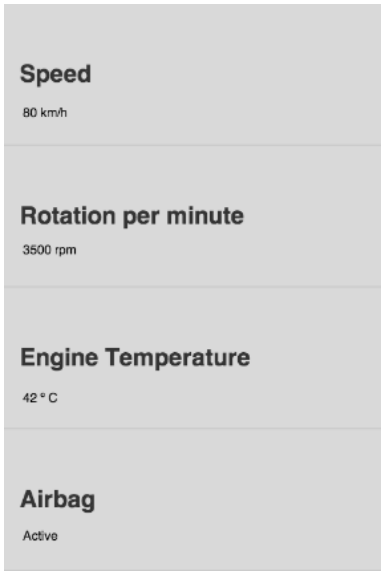


Figure 3.5: OBD-II layout mockup.

The navigation between these layouts should be intuitive. Different platforms provide different navigation types. At the implementation stage, the navigation types for the platform in which HDy Copilot is being developed, should be analysed and the most appropriate should be deployed in the application.

Another layout that should also be implemented is the settings. This layout is depicted

in figure 3.6 and should be displayed to the user when this needs to introduce contact info and general preferences data, to be used by HDy Copilot.

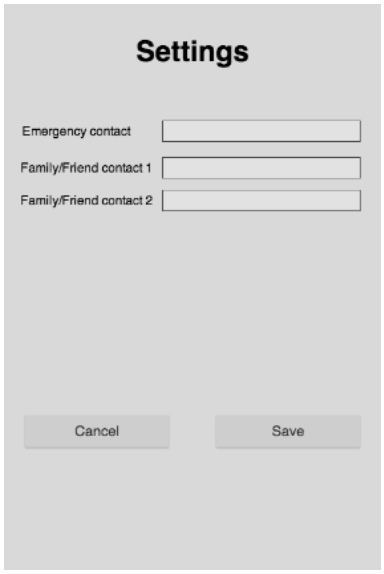


Figure 3.6: Settings layout mockup.

3.3.4 Requirements

To be successfully integrated as an AU and fulfil the aimed goals, HDy Copilot should fulfil a set of functional requirements to assure the robustness, usability safety and performance. These requirements were already mentioned directly or indirectly throughout this document and this section intends to summarize them. The requirements are the following:

- Communication with HEADWAY;
- Background execution;
- User info and preferences persistent storage;
- Display information safely to the driver;
- Accident detection algorithm;
- RHW manual report;
- Display OBD-II real time data;

As soon as HDy Copilot is launched the communication session should be established and information may be exchanged. HDy Copilot should be capable of working in background, since the AU is a smartphone, other application can be working in foreground. The user should be able to save and access his preferences, such as friends and family contacts to notify in case of accident, among other data.

Data frames to be exchanged with the OBU should be previously agreed and defined. Incoming RHW should be automatically displayed safely to the driver.

The ADA requires access to the device's sensors and location as well as OBD-II data as it was referred in section 3.3.1. The algorithm must transmit the necessary info to the OBU when an accident is detected. It should also send a notification to the driver's family and friends contacts stored in the database as well as to perform a help request.

HDy Copilot should also allow the driver to visualize he's vehicle sensor data in real time.

3.3.5 Overall system integration

To finalize this proposal its important to provide a general overview of the AU integration with HEADWAY. Its important to understand that the Application layer is executed in the embedded Linux Computer, and its responsible for communication with the smartphone device. The AU is an extension of the Application layer and should be tightly integrated with it. The communication session should be initiated both on the smartphone and the HEADWAY's Application layer and data formats should be well specified in both sides. The AU is responsible for detecting RHW and providing the related data to the OBU, to be processed and generate a DENM. Figure 3.7 depicts the purposed integration.

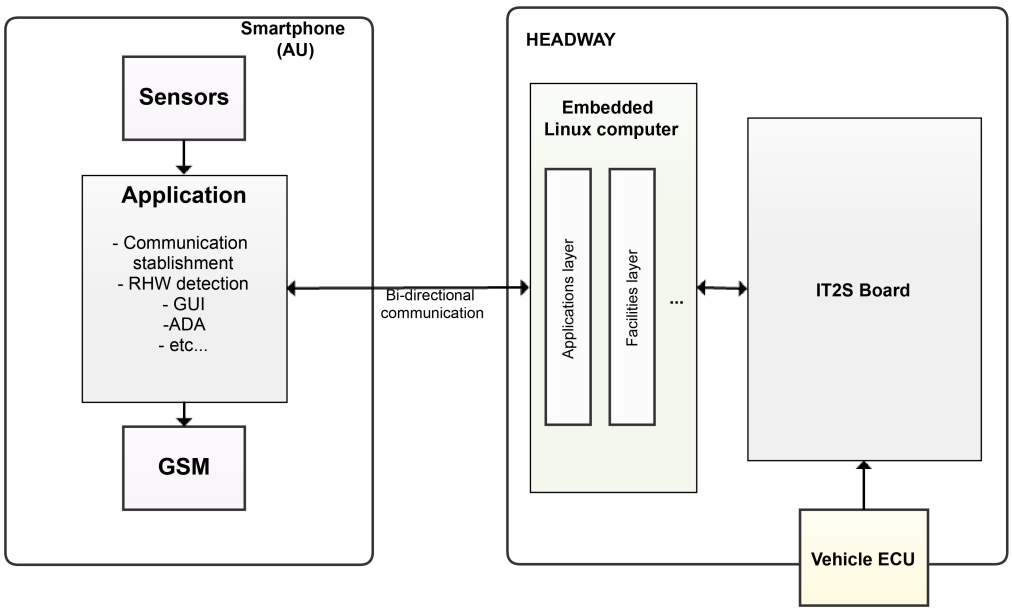


Figure 3.7: Targeted implementation.

Summarizing, it's important to retain from this chapter that the smartphone is supposed to be integrated with HEADWAY. The proposal is based on a smartphone application to be integrated with the HEADWAY that would act as a HMI by implementing a GUI along with the RHW report feature. Another feature to be implemented in the application, is the ADA, which would deal with the AAD and the eCall generation aspects.

Having the proposal formulated, its possible now to proceed to the development stage. The next chapter describes the development process and choices.

Chapter 4

Implementation

This chapter describes HDy Copilot's implementation. The decisions made along the implementation are explained and always based on the concepts presented in the chapter 2 and the proposal presented in chapter 3.

The chapter starts by describing the choices of the development environment for HDy Copilot, the device used and how it's integrated with HEADWAY, particularly the communication interface, the data to be exchanged and the frame formats.

HDy Copilot's implementation description is separated into two different sections, one regarding the layout design and other regarding the application features and functionalities (core). In this last section HDy Copilot's behaviour and structure is described and the ADA implementation is detailed.

4.1 Mobile platform

In section 2.3 platforms were analysed and an assessment was made to verify which platforms are more commonly used by smartphone users. Android and iOS turn out to be the most relevant of them all. Also section in 2.3, two types of development are described. Native and Multi-platform development, both providing advantages and setbacks in application development.

It was concluded that, Native development offers higher performance relatively to Multi-platform. As chapter 3 describes, HDy Copilot should implement an ADA that uses sensor data. Also information is supposed to be exchanged between the HDy Copilot and HEADWAY. As this is a safety and efficiency application, high performance is required to provide quick and reliable responses.

The use of Multi-platform development tools, provides the capability to execute HDy Copilot in different platforms. The goal of this dissertation is to develop a prototype and the fact that it can only be deployed in a smaller range of devices is not a concern.

With these premisses, native application development was the choice made for HDy Copilot development. When developing natively, the application can only be executed on the targeted OS. Android was the choice made. This platform is open-source and doesn't offer connectivity limitations like the MFi program does for iOS. It also provide all the tools necessary to develop HDy Copilot features.

HDy Copilot was developed for Android 4.2.2, currently the most recent release of this OS. The device used was a Samsung Galaxy Nexus, provided by IT-Aveiro. This device possess

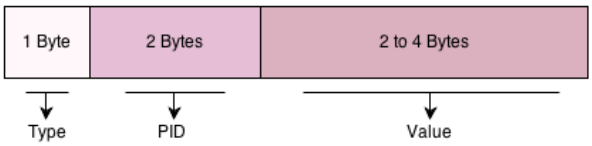
all the hardware resources necessary for HDy Copilot's development.

4.2 Communication interface

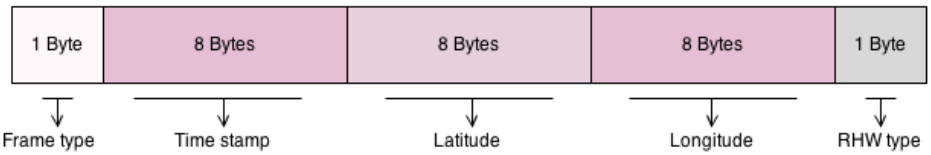
HEADWAY provides two means of external communication, namely the USB and the Bluetooth. The Android device to be integrated with the system as an AU, needs to constantly exchange data. Two types of data are exchanged between the Android device and HEADWAY, namely the vehicle sensor data and the RHW data. The vehicle sensor data (OBD-II) is exchanged in one direction only, from the vehicle to the device. The RHW can be transmitted by both the device (RHW manual report, or AAD) and the vehicle (incoming RHW).

HDy Copilot's main functionalities rely on this data exchange. The developed accident detection algorithm can use the vehicle airbag deployment signal to detect accidents. The communication interface must be the most reliable possible. Adding to this, since the ADA requires several sensors and the device GPS location system, the device power usage is a concern. Due to these facts the USB was chosen for the connection interface. This communication type, offers great reliability and provides energy to the device, eliminating the power usage problem.

As mentioned before, two types of data are exchanged between the device and HEADWAY. The RHW and the OBD-II (vehicle sensors) data. To manage this data, two data frames were agreed by the both communication interventionists. Figure 4.1 depicts both data frames.



(a) OBD-II data frame



(b) RHW data frame

Figure 4.1: Agreed data frames.

As presented in the figure 4.1a the OBD-II data frame is composed by a frame type followed by the Parameter Identificator (PID) and the value. The frame type indicator is encoded with one Byte and signalises the type of data carried by the frame. The OBD-II signal PID field is encoded by two Bytes and indicates the OBD-II signal (speed, airbag, etc.). The following data field, the value, contains the value/quantity of the signal. For the majority of the signals two or four bytes are needed to encode this data.

Figure 4.1b presents the RHW data frame. As mentioned before, this data frame can be transmitted by both HEADWAY and the Android device. Like the OBD-II data frame, this one contains a frame type indicator encoded with one Byte, followed by the RHW time

stamp, latitude, longitude and type. The time stamp, latitude and longitude are encoded with eight Bytes each. The frame is finalised by the RHW type encoded with one Byte.

The USB connection provides the Android with 500 mA. This current allows the device to maintain its charge even when running the ADA.

4.3 HDy Copilot's implementation

As mentioned in 4.1, Android was chosen for HDy Copilot's native development. This decision implies that Android guidelines, described both in appendix A and appendix B, were followed in HDy Copilot's development.

Application development in Android can be divided in two main work fields:

- Design - Related to the GUI elements, such as icons, colors, layouts, images and visual effects.
- Core - Related to the application's functionality. It allows the GUI elements to perform tasks when demanded as well as other features that the developer introduces to the application.

The two work fields are strongly related, but will be described separately. In each work field description, several references can be made to the other as a result of their relationship.

4.3.1 Design

The GUI design, influences how the application works and dictates the interaction between the user and the application. The application design differs from OS to OS. As mentioned before, Android was chosen for development. This OS has is own design language and guidelines (described in appendix B). HDy Copilot's design follows the GUI mockup, presented in section 3.3.3 as a proposal. Adding to the mockup, the Android design guidelines were followed to produce intuitive layouts for Android and non-Android users. Other layouts were created at the development phase, that appeared as a necessity and were not predicted in the proposal stage. Due to this fact, their mockup was not presented in chapter 3.

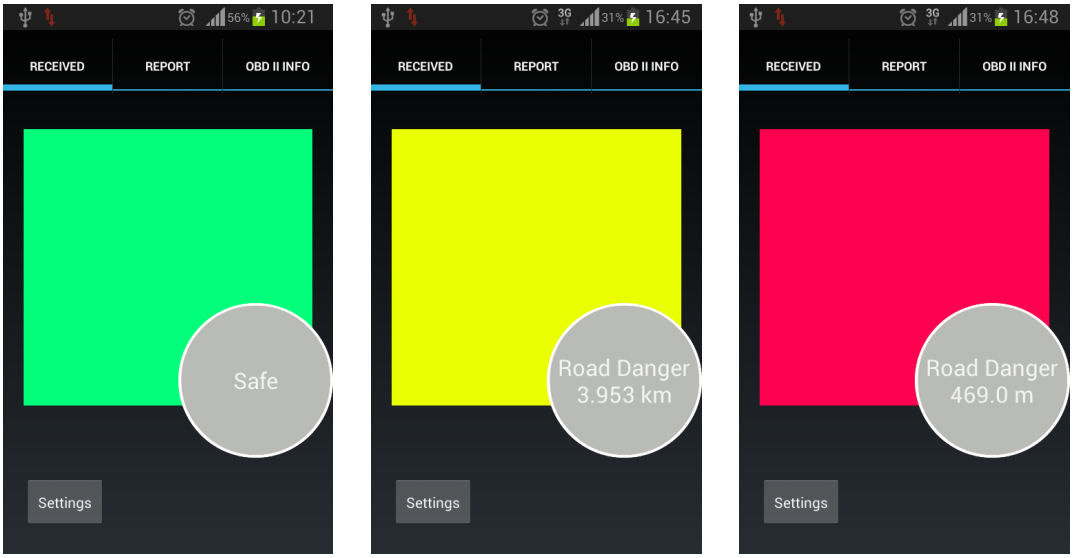
When designing HDy Copilot GUI, the main goal was to keep it simple, easy and superficial. This allow minimal touch interaction and therefore draws less attention from the driver. To keep it simple and straightforward the navigation type implemented was tabs, recommended by the guidelines, described in appendix B, for superficial applications. Three tabs were implemented to integrate the Receive, Report and OBD-II activities[1]. With tabs, the driver can quickly change layouts and achieve his goal.

The first tab (to the left), contains the Received activity. Its layout followed the GUI mockup proposal in section 3.3.3 and is depicted in figure 4.2.

As figure 4.2 depicts, the Received activity layout is simple. It's composed by a large square that can take the colors green, yellow and red, by a grey circle that contains textual information and by a Settings button. The Received activity was developed to warn the driver of hazardous events (RHWs) that occurred ahead on his route, so that he can approach the location with caution. The layout was designed to be simple and to enable the retrieval of information in short time (visual glance). The square represents a visual warning that changes

[1] Android activities are described in appendix A

color according with the distance between the driver and the RHW received. If no notification is received, the square takes the color green, indicating that the route is safe in that moment. When a RHW is received at a considerable distance, the color changes to yellow, or to red if the distance of the RHW is short. On each notification, additional information is displayed, in the form of text, in the gray circle. This information is composed by the received RHW type and its distance to the driver. When there is nothing to report the circle will contain the word "Safe". The last visual element present is the Settings button. Android guidelines advise the use of a menu button, for settings and other options, located in the action bar. The action bar is a component of the typical Android application design (described in appendix B). The decision to create the Settings button outside of the action bar, was supported by the need to clear the visual appearance of HDy Copilot's layout and to avoid miss taps[2] Because of this, the button was drawn on the bottom left corner of the layout and the action bar was hidden to clear the visual and increase the available screen real estate. The button's dark grey color is intended to avoid attracting the driver's attention when driving.



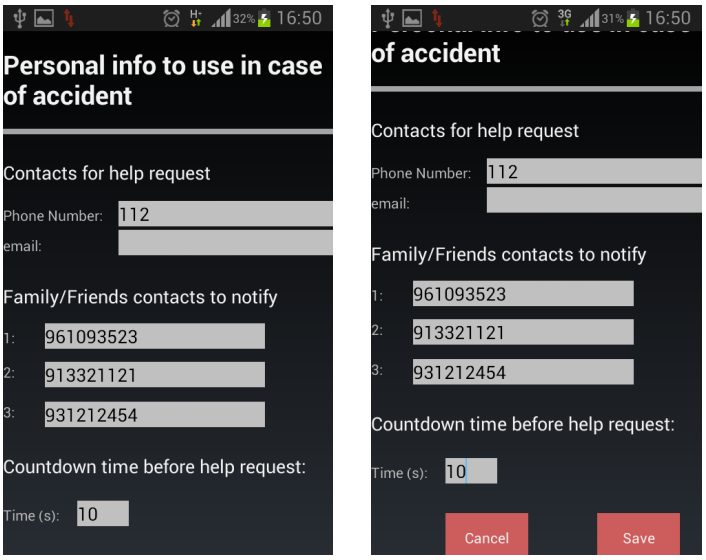
(a) No warning. (b) Long distance warning. (c) Near by warning.

Figure 4.2: Received tab layout.

The Settings button, when tapped, launches the UserSettings activity, which has the layout depicted in figure 4.3.

[2]Tap is an Android gesture. Android gestures are discribed in appendix B.

The UserSettings activity was developed to allow the user to save his family or friends contacts, that he wishes to be notified in case he suffers an accident, the emergency contact (EMS) for the eCall and the countdown (explained in section 4.3.2) time in seconds. In its layout, apart from the textual information and the editable fields, there are also two buttons. The Cancel button closes the UserSettings activity without saving data. The Save button closes the UserSettings activity and saves the preferences, displayed on the editable fields, into a database. This process is also explained in section 4.3.2.



(a) UserSettings layout scrolled up.

(b) UserSettings layout scrolled down.

Figure 4.3: UserSettings layout.

One of HDy Copilot's main features is the eCall system, which has the ADA at its core. To execute this algorithm the GPS location is needed. If the GPS location system is not enabled, the user is prompt with a dialogue box, depicted in figure 4.4. This dialogue box contains two buttons that perform different tasks as described in section 4.3.2.

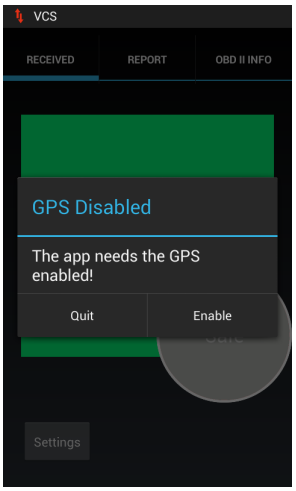
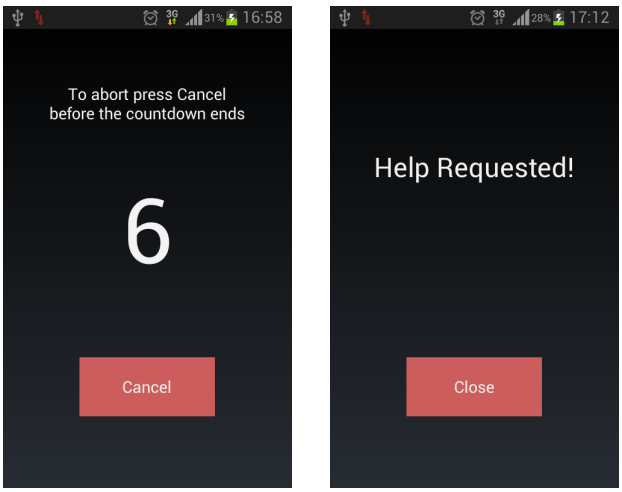


Figure 4.4: GPS enable dialogue box

If the ADA detects an accident, the Countdown activity is launched. This activity is intended to give the user some time to abort the procedure followed by the algorithm upon accident detection, thus avoiding a possible false detection. The Countdown activity layout is depicted in figure 4.5. Figure 4.5a presents the countdown timer and the Cancel button. If tapped, the Cancel button closes the activity and aborts the procedure. If the countdown is not interrupted, the algorithm proceeds, as described in section 4.3.2, and the layout changes, by informing the user that help was requested. This is depicted in figure 4.5b.

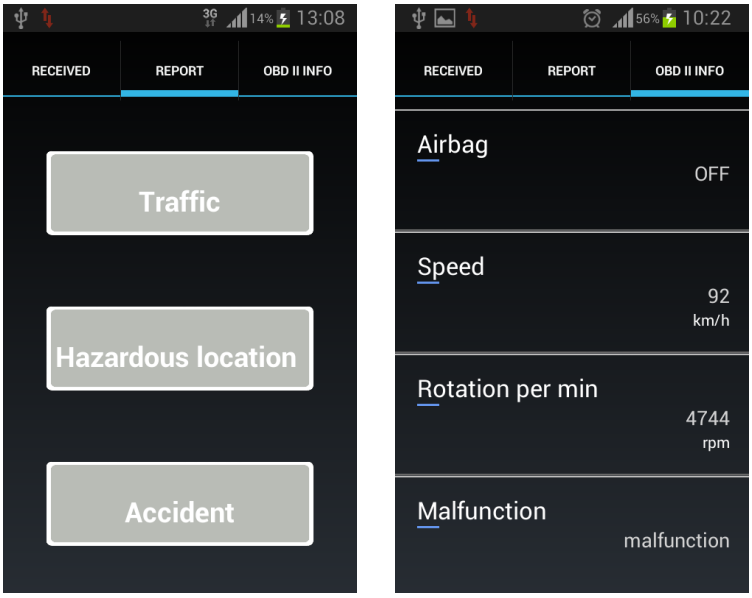


(a) Countdown timer. (b) Help requested.

Figure 4.5: Countdown activity layout.

The Report activity is also integrated in the tab navigator (middle). This activity was developed to allow the driver to report RHWs to other vehicles throughout HEADWAY. Its layout is depicted in figure 4.6a. It follows the mockup proposed in section 3.3.3 and depicted in figure 3.4. This layout contains three large light gray buttons with white textual information. The textual information indicates the type of RHW the button will report. The color chosen for the buttons allow a high contrast, in order to be easily visible during the driving act.

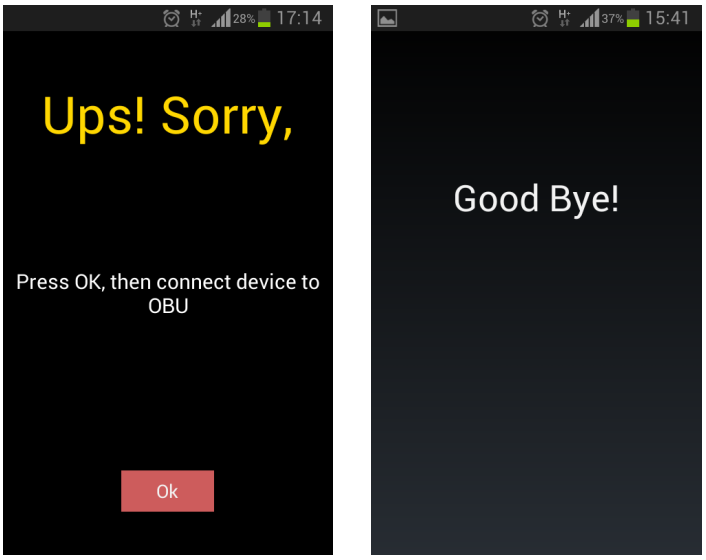
The last tab (right side) integrates the OBD-II activity. Its layout presents the vehicle sensor information, intended to provide the driver the ability to check his vehicle status. The layout design follows the mockup proposed in section 3.3.3 and presented in figure 3.5. Its composed by a list that contain the name of the sensor, its actual value and measurement unit. The layout is depicted in figure 4.6b.



(a) Report layout (b) OBD-II layout

Figure 4.6: Tab layouts

There are still two extra layouts designed to hint the user what is happening in two distinct situations. These layouts are the Warning layout and the Goodbye layout, both belong to the MainActivity and are depicted in figures 4.7a and 4.7b respectively. The first is intended to notify the user how to proceed, when he tries to launch HDy Copilot without connecting the device to HEADWAY. The second is intended to hint the user that HDy Copilot is purposely closing due to his actions.



(a) Warning layout (b) Good bye layout.

Figure 4.7: Extra layouts

4.3.2 Core

Software development is usually composed in four phases. Requirements identification, design, development and testing. There are several processes that can help a software developer in each of these phases. The first phase was already described in section 3.3.4. This section is about the software design. Design here, should not be interpreted as graphical design (draw), it should be interpreted as the software architecture and how different components are developed and interact one with another.

This section describes HDy Copilot's software structure and behaviour. HDy Copilot is targeted for Android OS, therefore its programming language is Java, which is object-oriented. To describe HDy Copilot's software design, the Unified Modeling Language (UML) [41] was used. UML is a family of graphical notations that help in describing and designing software systems built using object-oriented styles. According to [42], UML is composed by thirteen different diagrams that can be group by:

- Structure diagrams - Depict system elements that are independent from the time instant. To this group belong the Class, the Composite Structure, the Object, the Component, the Deployment and the Package diagrams.
- Behaviour diagrams - Depict the system behaviour. To this group belong the Activity, the Use Case, the State Machine and the Interaction diagrams.
- Interaction diagrams - This is a subset of the Behaviour diagrams that depict object interaction. To this group belong the Sequence, the Communication, the Interaction Overview and the Timing diagrams.

To describe HDy Copilot's software design three diagrams are used, the Use Cases, the Activity and the Class diagram. Along side with these UML diagrams is a textual description and in some cases flow chart diagrams. Note that the UML Activity diagram activities are distinct from Android activities.

Behaviour

Software development is about specifying rules and small tasks to be performed by a machine, in a certain order, to achieve a pre-determined goal. A software developer before starting to write code (developing), first needs to design the all project in order to asses its needs as well as organize his (or the team) work, so that the final project is delivered on time and close to the initial idea. To specify the application software design UML was used. As mentioned in section 4.3, UML can be used to specify and describe the software behaviour as well as its structure (static behaviour).

UML has thirteen diagrams, however not all of them must be used to specify and model a project. UML diagrams should be used according to developer's necessity. To describe the HDy Copilot's behaviour two diagrams were put to use, the Use Cases diagram and the Activity diagram.

As the application is defined and its requirements enunciated in section 3.3.4, a first visual scheme of the application behaviour is presented in figure 4.8, with a Use Cases diagram. With this diagram is possible to understand how the system interacts with the user as well as with other systems.

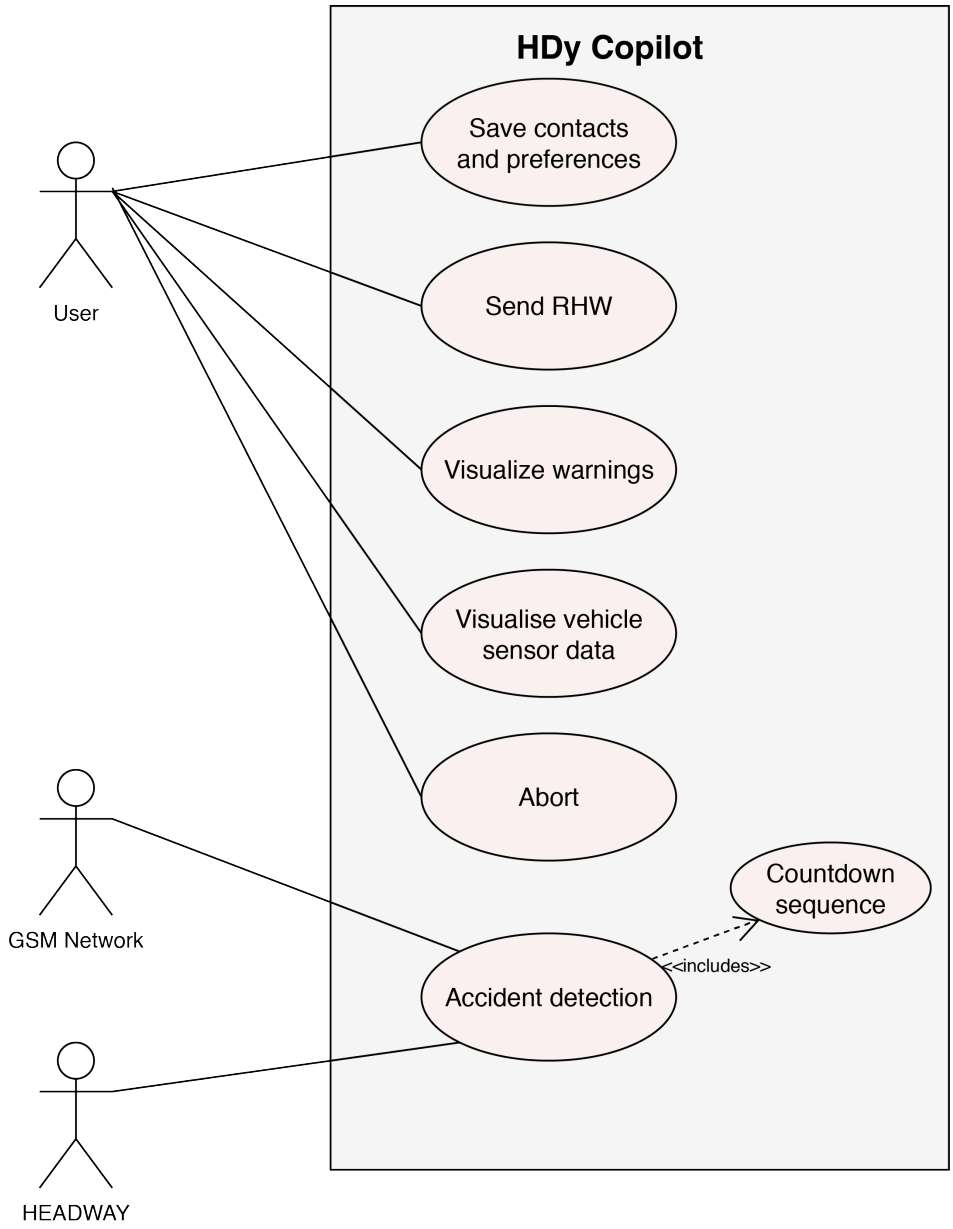


Figure 4.8: HDy Copilot's Use Cases diagram.

Observing the diagram its possible to verify the existence of three actors. The user, HEADWAY and the GSM network. The red coloured ellipses represent the use cases. The connection between the use cases and the actors represent the relationship and interaction between them. The main actor is the user. There are five uses cases for this actor. The driver can interact with the application by setting and saving his contacts info and preferences, through the UserSettings activity, by reporting RHWs with the press of a software button, through the Report Activity, visualizing received RHW, through the Received activity, visualizing vehicle sensor data, through OBD-II activity and by aborting the help request performed by the ADA.

As depicted in the Use Cases diagram in figure 4.8, the User does not interact with the ADA directly. He can only abort its accident validation. The ADA however, interacts with the remaining two actors. It uses HEADWAY, to receive RHW and vehicle sensor data as well as to send RHWs. It also interacts with the GSM network by sending SMSs to the user family/friends contacts and by performing the eCall.

The Use Cases diagram provide an idea of how HDy Copilot behaves, especially regarding its interaction with the user and other systems. To describe how HDy Copilot behaves in a deeper perspective, an Activity diagram was elaborated. This diagram represents graphically HDy Copilot's work flow, i.e, the procedures resulting from the system functionalities and its interaction with the user and other actors.

In the Activity diagram the rounded objects are called actions. These actions define how the application deals with certain situations and together they describe the work flow of the application. The square object is a note. In this type of diagram parallel actions can be presented through the fork/join elements. HDy Copilot's Activity diagram is presented in figure 4.9.

The application launches as soon as the Android device is connected through USB to HEADWAY. In Android there is always an activity responsible for launching the application. In this case its the MainActivity. Once this activity is created[3], it starts a service[4], named USBService, to establish a USB session. If this session initiation fails, the application displays a warning screen, depicted in figure 4.7a. If the session initiation succeeds, the application starts exchanging data with HEADWAY.

When the application launches, the MainActivity creates the Received, Report and OBDII activities and instantiates some of the required data, such as sensors, location systems, etc. It also proceeds with the Preferences action. This action has a rake symbol on its right hand side, which indicates that it has a sub Activity diagram. Sub Activity diagrams are used to simplify the diagram. This action's sub Activity diagram is depicted in figure 4.10.

[3]Activities are created through the onCreate() callback method described in appendix A.

[4]Android services are described in appendix A

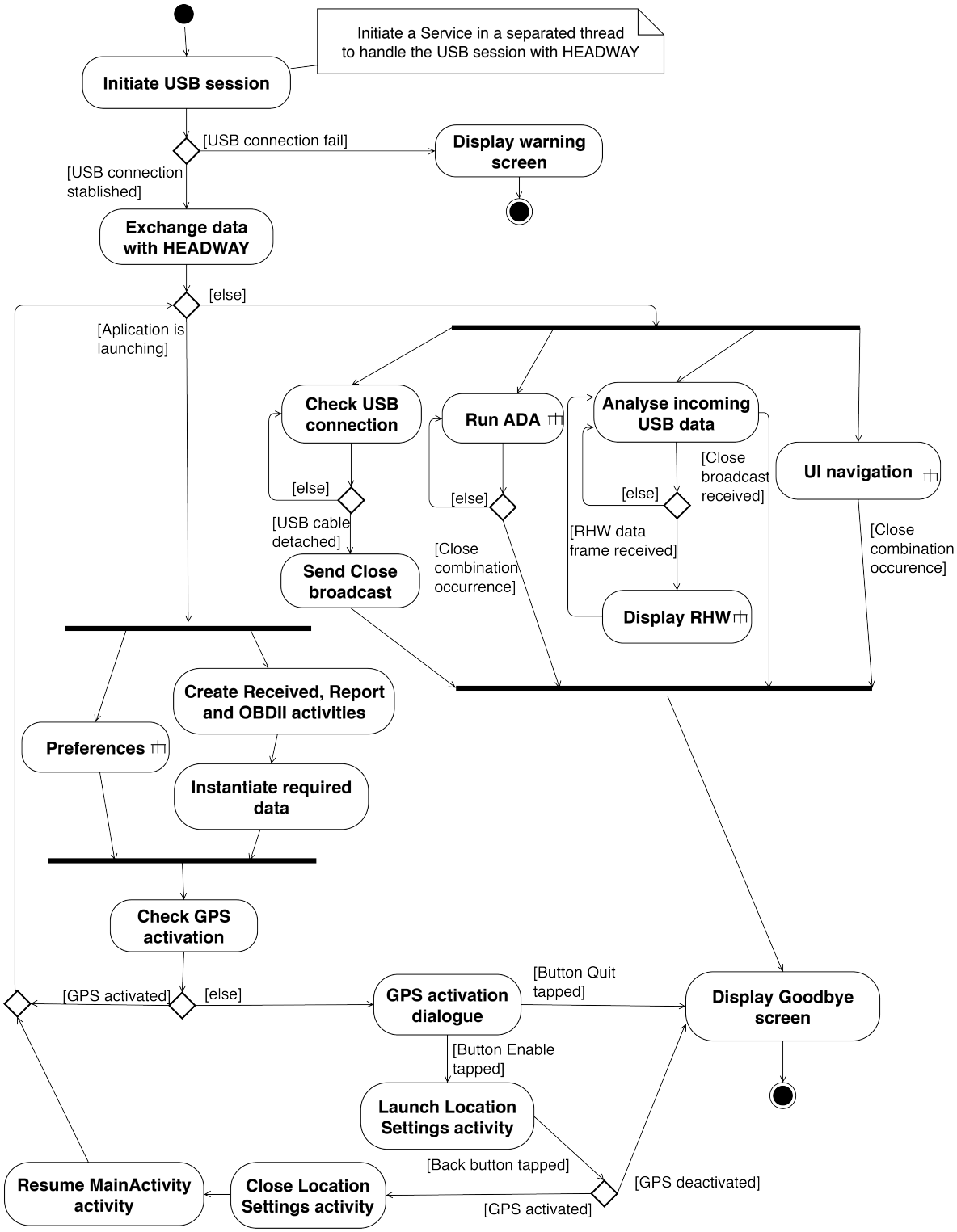


Figure 4.9: HDy Copilot's Activity diagram.

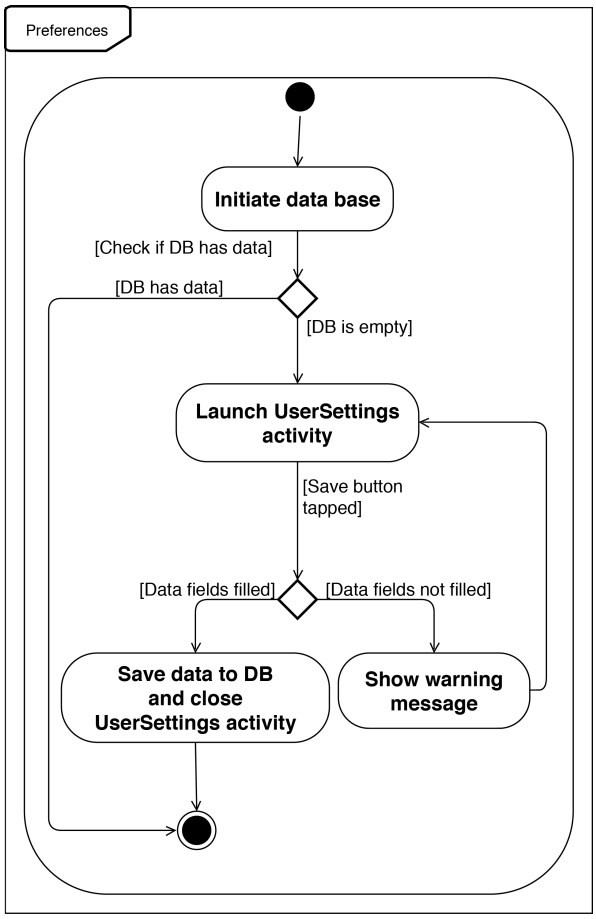


Figure 4.10: Preferences sub Activity diagram.

As figure 4.10 depicts, if the database is empty (happens when the application is launched for the first time) the UserSettings activity is launched. This activity has the layout depicted in figure 4.3. When the user taps the Cancel button, nothing happens. When the user taps the Save button, a check on the data fields is performed. If the data fields are empty, a warning message is displayed and the activity remains opened. If the data fields are correctly filed, the data is saved to the database, the UserSettings activity is closed[5] and this action is finished.

Since most devices possess a GPS location systems, that provides a fairly accurate location data, HDy Copilot uses the GPS for its ADA, for the RHW manual report and for calculating the distance between the user and an incoming RHW. If the GPS is not activated, the user is prompt with a dialogue box, as depicted in figure 4.4. At this moment the user has two choices, to press the Enable button, which leads to the Location Settings activity (activity not belonging to the application), or the Quit button, that leads to the presentation of a goodbye screen, depicted in 4.7b followed by the application termination. If the user is at the Location Settings activity, he can activate the GPS. When he taps the device back button to navigate back into HDy Copilot, a verification is again made, and if the GPS is not activated

[5]Activities are closed through the callback method onStop(), described in appendix A

the application will present the goodbye screen and terminate. If the GPS is enabled the Location Settings activity closes, the MainActivity is resumed and the user is presented with the application GUI.

At this moment the application launch state is over and its not on the execution state. The ADA is initiated and starts executing. The Run ADA action also has a rake symbol on the right hand side. For simplicity reasons, this action sub Activity is described later and depicted in figure 4.15.

Data is being exchanged with HEADWAY all the time. OBD-II signals and RHWs are the two types of data exchanged. The incoming data frames are constantly being analysed and in the case a RHW data frame is received, its information is displayed to the user. Once again note that the Display RHW action has a rake symbol on its right hand side. Its sub Activity diagram is explained later on this section and is depicted in figure 4.12.

Another important aspect of the application work flow, is the user interaction with it. This interaction is bundled in the UI Navigation action. This action sub Activity is depicted in 4.11.

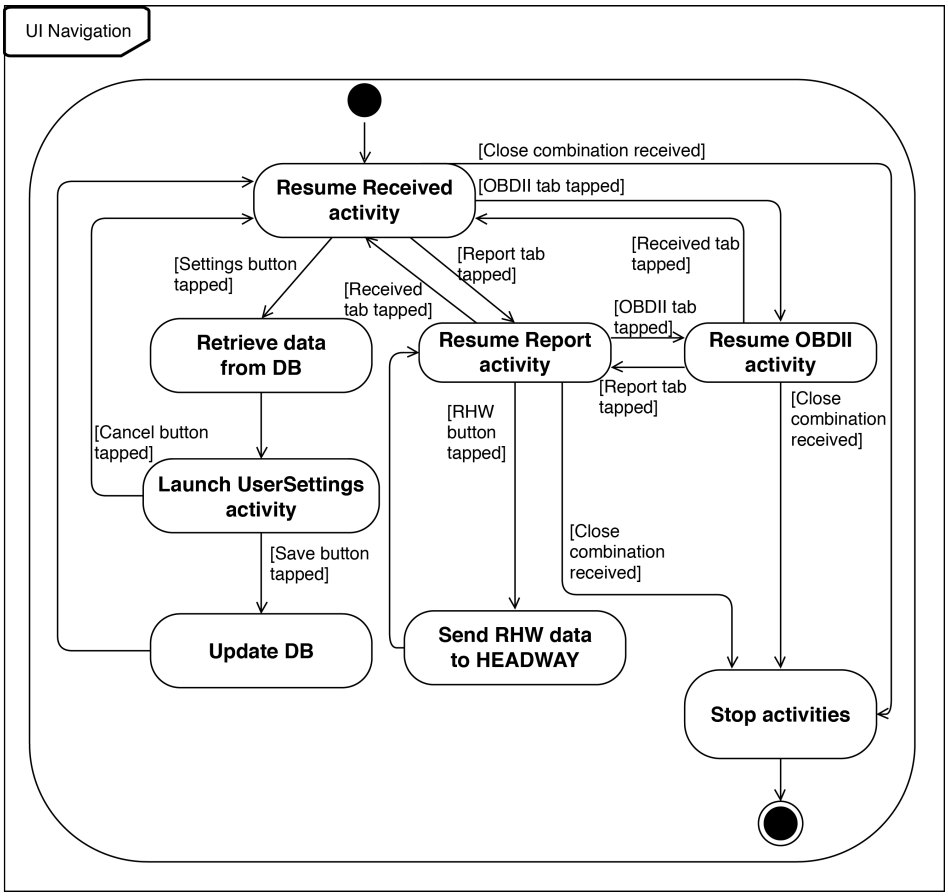


Figure 4.11: UI navigation action sub Activity diagram.

As mentioned before, after the application launch, the Received tab is automatically selected. The user here can navigate the application. In this activity layout, as depicted in figure 4.2, is a Settings button. When tapped, the data from the database (DB) is retrieved

and the UserSettings activity is launched. Here the user can edit his preferences and contacts. When the Save or Cancel buttons are tapped, the activity closes and the application returns to the Received activity, updating the DB in case the Save button was tapped.

To navigate the application the user can use the tab navigator. This navigator has three tabs, as depicted in figures 4.2, 4.6a and 4.6b. If the user taps the Report tab, the Report activity is resumed. Here the user is presented with the layout depicted in figure 4.6a. This layouts presents three light gray buttons with a textual description of the RHW they represent. When the user taps these buttons, data such as current time stamp, location and RHW type is sent to HEADWAY.

From here, the user can navigate back to the Received activity, by taping the Received tab, or he can navigate to the OBDII activity, which will present him the layout depicted in figure 4.6b. Here the user can read vehicle sensor information.

Every activity has a registered broadcast receiver[6], which is used to receive announcements from other activities. The Android system itself generates broadcasts when certain situations occur, and in this case, the USBService broadcast receiver is ready to capture the USB cable detached broadcast, which is generated by the system when the USB cable is detached. Once this broadcast is received, USBService generates a close broadcast to be received by all other activities. When the close broadcast is received by the activities, all execution related to USB data exchange is ceased and the USBService is stopped. The ADA continues to be executed until the user tapes the back button. If the back button is tapped, the goodbye screen, depicted in 4.7b is displayed and the application terminates.

The application is terminated when the USB cable is detached and the back button is pressed, respectively. This combination of events is the only way for the user to terminate the application. With this combination the ADA assures accident detection event if during a collision the USB cable is detached.

As mentioned above, the Display RHW action has a rake symbol indicating a sub Activity diagram. This action is very important in the HDy Copilot. As proposed in chapter 3, HDy Copilot should deliver the RHW notifications to the driver (user) in a way that don't require the driver loss of focus on the road. The Received activity layout was designed to allow a quick interpretation of the event, but its not enough. As described in appendix A.1.1, only one activity can be resumed at a time, meaning that only one activity layout can be displayed at a time. This raises concerns on the Display RHW action. The RHW should be displayed to the driver without he's need of interacting with the device. The solution deployed is presented in the sub activity diagram depicted in figure 4.12.

As described in section 4.2 and depicted in figure 4.1, exchanged data is organized into specific data frames according with the info they carry. When an incoming RHW data frame is detected, its information is presented to the user. This process is presented in figure 4.12.

When a RHW data frame is received, its data, namely the time stamp, the location and the type, is extracted. Then the device GPS location is retrieved. With the device and the incoming RHW locations, the distance between them is calculated using the Harvesine formula [43]. This distance is used to decide which color the Received activity layout square takes. At this development stage the two kilometre threshold is used as a decision point. This value is merely for demonstration purposes and can be changed in the future. If the RHW is located at a distance greater or equal than two kilometres the square will turn yellow, otherwise will turn red. The distance and the type of RHW is printed on the gray circle of the

[6]Broadcast receivers are described in appendix A

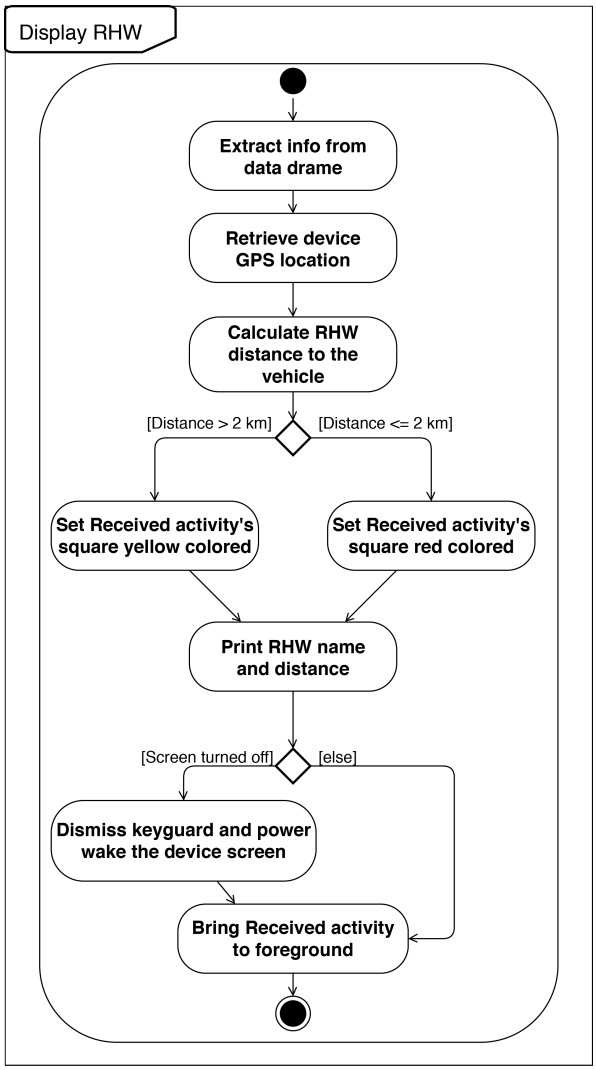


Figure 4.12: Display RHW action sub Activity diagram.

Received activity layout. After these alterations to the layout occur the application dismisses the keyguard and powers on the screen (if its turned off) and brings the Received activity to the foreground (resumed state). The user then, needs only to look at the info to interpret the RHW. The square will be turned green, and the circle will display the "Safe" text after the screen gets powered off again.

There is still one last action to be described in the HDy Copilot's Activity diagram. The Run ADA. This action is fairly complex, and its execution is based on the background research performed in section 2.5. Its implementation and sub Activity diagram are described in the following subsection.

Accident Detection Algorithm implementation

The ADA is at the core of the eCall system targeted for implementation and aims to provide the application with the means to automatically detect vehicle accidents. To detect most car accidents, collisions and rollovers should be detected. The algorithm implemented handles both situations.

As mentioned in section 2.5.1 car collisions produce certain acceleration values. These acceleration values can be used to predict the severity of injuries inflicted on the car occupants.

To detect collisions, the algorithm requires the linear accelerometer. This accelerometer output is not influenced by gravity. The device possess a three axis linear accelerometer. The linear accelerometer coordinate system is fixed to the device, i.e, it never changes as the device moves. No matter what the position of the device is, the axis are as depict in figure 4.13. The Z axis is perpendicular and point out to the device's screen. The Y axis goes from the bottom to the top of the screen and the X axis goes from left to right of the device when this one is positioned in portrait mode with the screen pointed to the user.

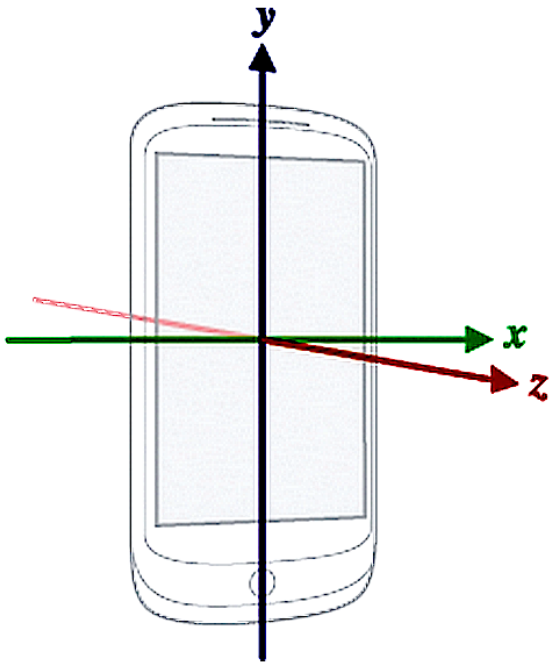


Figure 4.13: Android sensor axis. *Source: [17].*

To detect collision the 4g threshold is used along with the ASI metric. This content was referred in section 2.5.1. As mentioned in section 2.5.1, several authors describe similar AAD mechanism that use smartphone as well as other technologies to detect car collisions. The authors also assessed that the 4g threshold is indicated to filter false positives such as smartphone drops, harsh breaks and sudden accelerations. The ASI metric is used to measure accident severity in European road restraint systems. This metric is based on acceleration values and computed as described in section 2.5.1.

The algorithm was implemented using the 4g threshold to assess if a car collision happened and the ASI metric provide an estimate of its severity.

To properly detect a collision, the vehicle and the smartphone must possess a inelastic bound, so that the forces felt by the smartphone are the same as the vehicle. This way when the device captures an $\vec{a}_r$ higher or equal than 4g a collision is detected. The $\vec{a}_r$ is the resulting acceleration vector and is determined according to expression 4.1, where D_x, D_y, D_z are the acceleration felt by the device in figure 4.13 X, Y and Z axis respectively.

$$\vec{a}_r = \sqrt{D_x^2 + D_y^2 + D_z^2} \tag{4.1}$$

The smartphone should be place in the vehicle with the help of a smartphone car holder or a similar solution. Its axis should be aliened approximately with the ASI metric axis depicted in figure 2.7 in order that:

$$\begin{aligned} A_x &= -D_z \\ A_y &= D_x \\ A_z &= -D_y \end{aligned}$$

Note that the signal (orientation) of the axis is irrelevant both in expressions 4.1 and 2.3.

Another method used for detecting collisions is the airbag deployment. This information is available through the OBD-II signals transmitted by the OBU (HEADWAY).

As mentioned previously, the algorithm also detects rollovers. To detect rollovers the algorithm constantly monitors the smartphone orientation, since both the device and the vehicle are supposed to possess an inelastic bound.

According to [44] the orientation of the device can be retrieved using two different approaches. One approach uses the angles based on the accelerometer and magnetometer output. The accelerometer provides a gravity vector (pointing to the Earth's centre) and the magnetometer a vector that points north. With this two vectors its possible to determine the device's orientation. However, with this approach, the output of the magnetometer can include a lot of noise in the calculated position.

Another approach is to use the gyroscope sensor. This sensor output is far more accurate (lower noise). The downside of this sensor is the drift. The Android gyroscope sensor outputs the rotation speeds for the three axis. To determine the actual orientation (position) these speeds need to be integrated over time. This is achieved by multiplying the rotation speeds by the time interval elapsed between the current and the last sample. This results in a rotation increment. Each of these rotation increments are added to determine the device absolute orientation. The rotation speed provided by the gyroscope includes small errors. These small errors add up over time resulting in a constant slow rotation, named gyroscope drift.

To accurately determine the device orientation a technique called sensor fusion was implemented. The technique algorithm was retrieved from [44] and was directly applied here. Note that this algorithm was published under the MIT-Licence [45], which allow its free use.

Using Android SensorManager API, its possible to determine the device orientation though the SensorManager.getOrientation() method, using accelerometer and magnetometer data. The output of this method provides data with high frequency noise, because of the magnetometer. To remove the noise a low-pass filter is applied. On the other hand the sensor fusion algorithm retrieves data from the gyroscope. This data is then multiplied by the sampling interval to determine a rotation increment. The device orientation is the sum of the rotations. To eliminate the resulting drift, a high-pass filter is applied. The resulting orientation is the sum of the low frequency components from the accelerometer/magnetometer orientation and

the high frequency component of the gyroscope orientation. Figure 4.14 presents the sensor fusion flowchart.

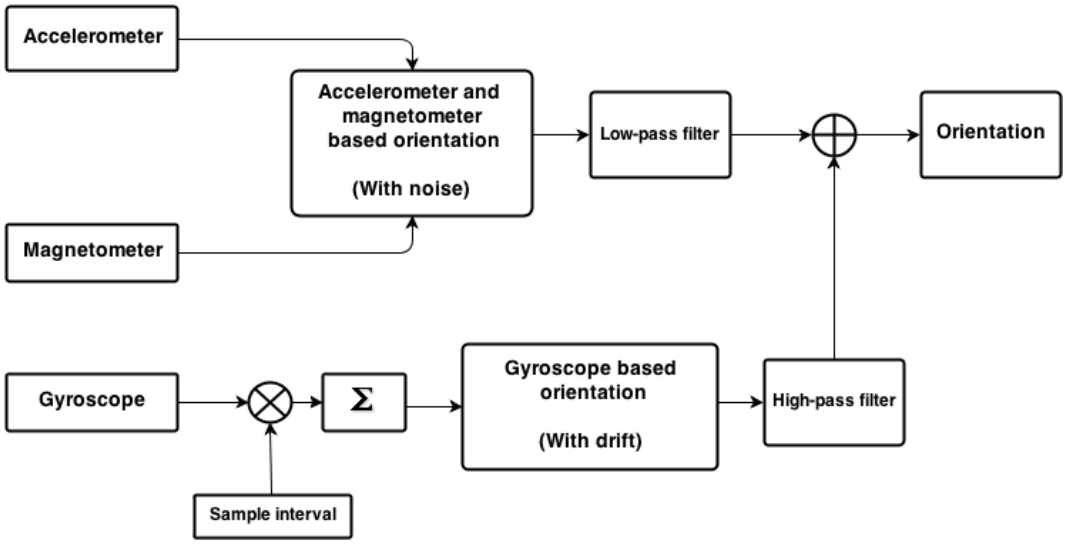


Figure 4.14: Sensor fusion. *Adapted from: [44].*

This technique outputs the X, Y and Z orientation changes, in degrees, relative to the initial position. The decision of rollover occurrence is made when the smartphone rotates at least 45 degrees over the device's Z axis, from its initial position. This rotation corresponds to the car rotation over the X axis (device Z axis) in figure 2.7. The angle that symbolizes the rotation over the device Z axis is named θ.

The Run ADA sub Activity diagram is depicted in figure 4.15. After the algorithm initialization, this one continues to be executed until the application in terminated.

The algorithm uses several Android device sensors. The sensors used are the following:

- Linear Accelerometer;
- Accelerometer;
- Magnetometer;
- Gyroscope;

In addition to these sensors the algorithm also uses the vehicle sensor data, particularly the airbag deployment signal. All this sensor data, along side with the user complacency by not aborting the accident validation is used to assess if an accident occurred as the sub Activity diagram depicts in figure 4.15.

Figure 4.15 presents the Run ADA sub activity diagram which provides a description of how the ADA is implemented. Once the algorithm is initialized, it constantly monitors three data sources, the incoming USB data frames, the required device sensors and the device GPS location updates, represented by the Analyse Incoming USB Data, Read Device Sensors and Request Location Updates actions respectively. The sensor fusion technique is applied at

each new sensor sample. The algorithm detects accidents when one of three situations occur, namely, the airbag deployment, rollovers and collisions.

As mentioned before, the algorithm is constantly analysing the incoming USB data, particularly, the OBD-II data frames. Once one of these frames is received, its data is extracted, and if it carries the airbag deployed signal, accident occurrence is validated and an accident detected broadcast is emitted.

The device's sensors presented above are constantly being analysed all at the same sampling frequency. This frequency is based on the requirements for the ASI computation described in 2.5.1. Initially sensors are calibrated. Then at each new sample the sensors are read and the fusion sensor technique is applied (except for the linear accelerometer).

When a new sample is outputted by the linear accelerometer, the $\vec{a}_r$ is calculated according with expression 4.1. If this value is lower than the 4g threshold, the process is repeated for the next sample. If the 4g threshold is surpassed, collision occurrence is validated and the algorithm proceeds to the calculation of ASI and the emittance a accident detected broadcast.

The sensor fusion technique initially, saves the device position (device initial position). If the position varies more than forty five degrees from the initial position, the device mean speed ($\bar{v}$) is checked. This mean speed is calculated using the GPS API, and is constantly being updated. If the mean speed ($\bar{v}$) is greater than 20 km/h, the process continues, otherwise it ignores the change in position and calculates it again. This mean speed verification was implemented to avoid false positives (false detections). The 20 km/h threshold is calculated based on the last ten seconds prior to the device position change. This threshold assures that the vehicle was moving before the rollover occurred. After the device position change, the algorithm checks the instantaneous speed ($\vec{v}_i$) for ten seconds. After an rollover, vehicle are usually immobilized. If that is the case the instantaneous speed should reach null. The threshold used is 5 km/h due to a verified GPS speed calculation inconsistencies at low speeds. If $\vec{v}_i$ is lower than 5 km/h, the algorithm validates the rollover occurrence and emits a broadcast, otherwise it repeats the process from the beginning.

Both the waiting time and the mean speed threshold are values used for demonstration purposes. This values are not permanent and can be changed, if they prove not be effective during real case tests.

After one of the three accident detectors validate accident occurrence, the algorithm proceeds to the retrieval of database stored informations and launches the Countdown activity. The countdown time is setted by the user in the UserSettings activity. As mentioned before, the Countdown activity layout, depicted in figure 4.5, enables the user to abort the process, by taping the Abort button. Aborting leads to the termination of this work flow. In the Activity diagram depicted in 4.9, its possible to verify that if a close combination is not received the algorithm is executed again. If the countdown is not aborted the algorithm proceeds with the help request.

The help request is made to three sources, The first is to the vehicular network, by transmitting a RHW data frame to HEADWAY. The second is to the EMS, by performing an eCall. During development it was verified that it was impossible to perform an eCall with the provided APIs. The solution found to surpass this problem was to send an SMS, containing the MSD, followed by a voice call to the EMS. In the SMS's content only the VIN is not included. This is due to the fact that it is still under assessment if in fact should be used, by the eCall responsible authorities, as well as its difficulty to obtain in the desired format. The SMS sent to the EMS is also sent to the stored friends/family contacts.

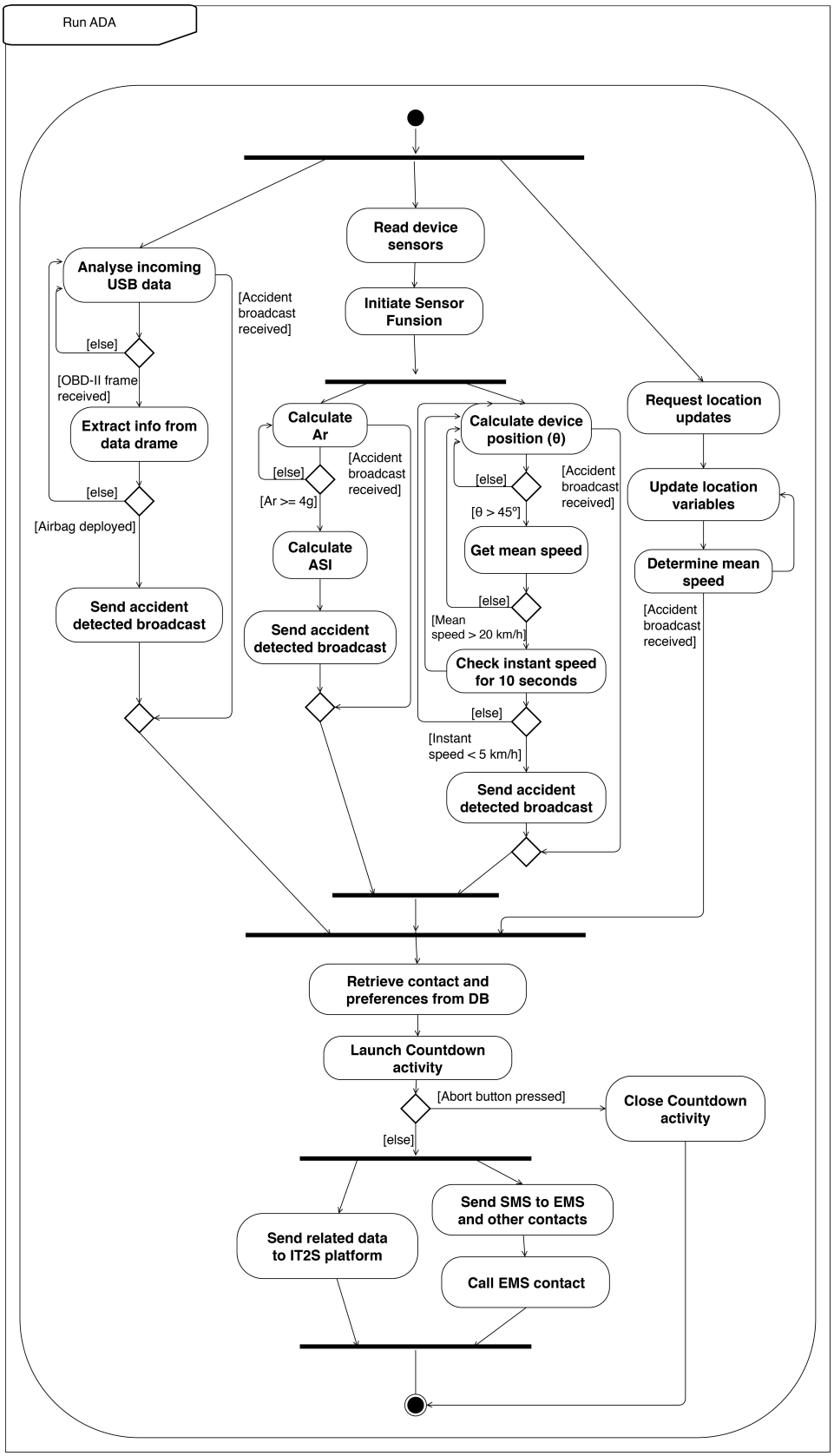


Figure 4.15: ADA sub Activity diagram.

Structure

HDy Copilot's dynamic behaviour is described by the Use Cases and the Activity diagrams. To understand how HDy Copilot was developed at its core, its important to have access to its static behaviour, i.e, its structure. To describe the HDy Copilot's structure a Class diagram is presented in figure 4.16. The attributes and methods of some classes were hidden to simplify the diagram.

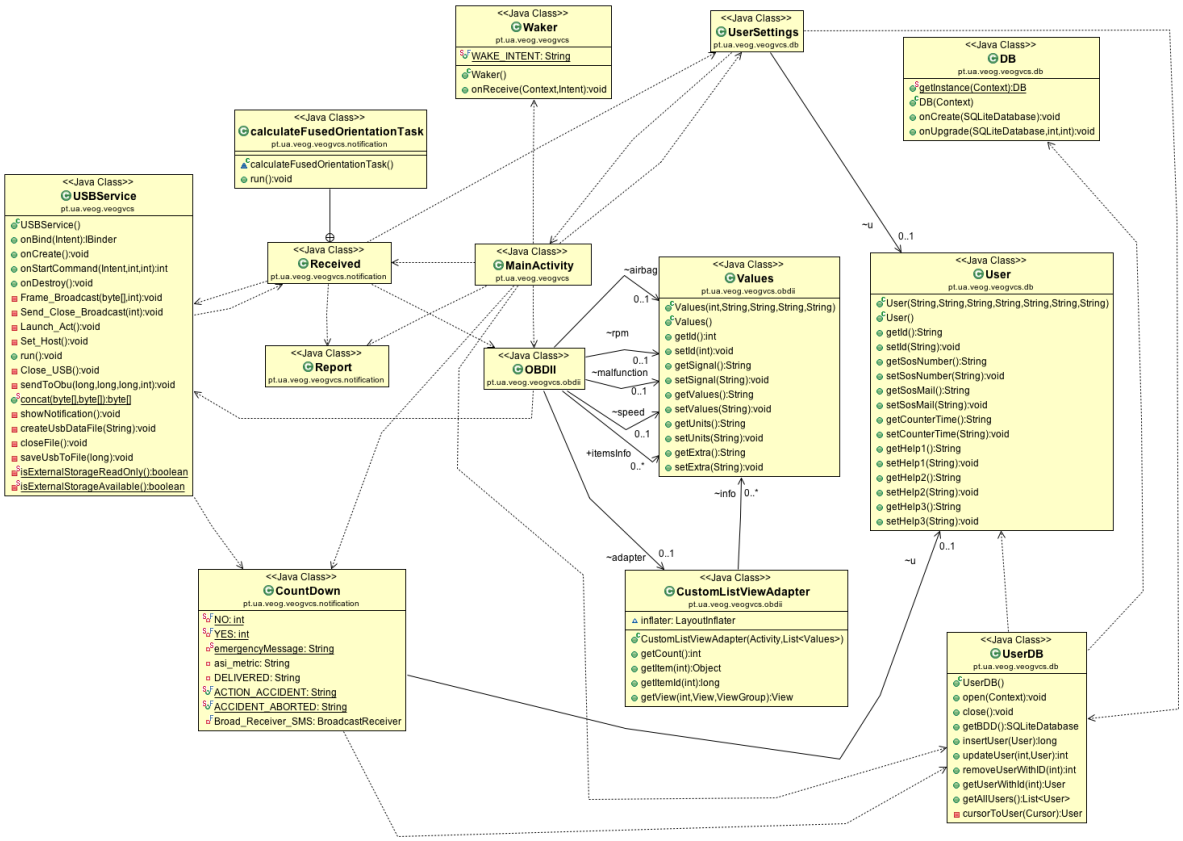


Figure 4.16: Class diagram.

HDy Copilot is composed by six activities named MainActivity, Received, Report, OBDII, Countdown and UserSettings, which all inherit the attributes, methods and life cycle from the superclass Activity, described in appendix A. The MainActivity is responsible for launching the application and most of the other activities as well as starting the service named USBService. This activity is also responsible for managing the application lifecycle. The UserSettings activity is used to access the database to save preferences and contacts introduced by the user through its layout interface. The OBDII activity is used to display and analyse incoming vehicle sensor data. The Report activity manages the RHW buttons. Here, when a button is tapped, the activity gathers and organizes the RHW related data to be transmitted to HEADWAY. The Received activity is responsible for displaying RHW to the user and determining its distance to him. Its also responsible for accessing the sensors, used in the algorithm as well as the device GPS system. The $\vec{a}_r$, the ASI and the θ calculations are performed here. The Countdown activity is launched by the Received activity once an accident is detected. This activity enables the user the option to assess the accident detection

validity, as well as it deals with the telephony (GSM network) aspects (SMS and voice call) of the help request.

The USBService inherits the attributes and properties of the Service superclass. This service is used to manage the USB connection and its data exchange.

Other objects are also part of HDy Copilot's structure. The object Values is responsible for encapsulating vehicle sensor data to be displayed in the OBDII activity's layout. This data is presented in the form of a list, and the CustomListViewAdapater object bind the data contained in the Values object to the list layout. The DB object creates a SQLite database table. This database is accessed and manipulated by the UserDB object. The data to be stored or retrieved to/from the database is encapsulated by an object named User.

Finally the Waker object is responsible for dismissing the device key guard, powering the screen and bringing the Received or Countdown activity to foreground, if the user receives a RHW or if he suffers an accident respectively.

As mentioned before, the activities and the service communicate between themselves through broadcasts, which are captured by broadcast receivers.

From this chapter is important to retain that the OS chosen was Android, which allowed the use of the USB connection to integrate the application to HEADWAY. The GUI design and the features of the application were both described in a behavioural and structural perspective. The ADA, which is one of the main features of the application, implements the eCall system, meaning the ADA and the help request. The eCall help request was not possible to implement, and, therefore an alternative was developed, based on voice call and SMS.

Having the application developed, it's possible now to proceed to the testing phase, to validate its functionalities. The next chapter presents the tests performed and its results.

Chapter 5

Tests and validation

After the HDy Copilot's implementation, there is the need to test it, in order to understand if the goals were achieved and if the system actually works as expected.

This section is about presenting some tests performed to the application, made to assess its robustness and its limitations. During tests, data was saved in a file in the device external memory. This data was then analysed with Matlab. The results are textually described and aided by graphics and tables containing valuable information.

5.1 Accident Detection

One of the most important features of HDy Copilot's is its AAD mechanism, from the eCall system, implemented by the ADA. The algorithm was developed to detect car accidents in the form of collisions and rollovers. Several tests were performed to assess if the algorithm and the device itself can detect both types of accidents. Airbag deployment signal tests were not tested due to its straight forward implementation. The tests were made only to the accelerometer based accident detection and the sensor fusion technique.

Due to the lack of an appropriate equipment to test the application in a controlled environment, the tests performed were improvised and therefore limited.

5.1.1 Collisions simulations and results

As described in section 4.3.2 the ADA detects collisions using the airbag OBD-II signal and the linear accelerometer built in the Android smartphone. This accelerometer outputs accelerations felt by the three axis without the influence of gravity. To validate an accident, the resulting acceleration ($a\vec{r}$) must be equal or greater than 4g. The first test performed was to assess if the smartphone linear accelerometer could sense accelerations at this magnitude. To achieve elevated accelerations, great changes of speed are required. This changes of speed are difficult to achieve without damaging the device. To address this issue the acceleration values were simulated by shaking the device violently using the hands. The results are depicted in figures 5.1, 5.2 and 5.3.

Figure 5.1 presents the $a\vec{r}$ value over the sampling time. The sampling time is the time stamp in UTC of each new linear acceleration sensor sample at the moment of this test. The pulse maximum value reached in this first test is $a\vec{r} = 4.441$g. This proves that its possible to surpass the 4g threshold, which indicates that the device can detect collisions. Still in this

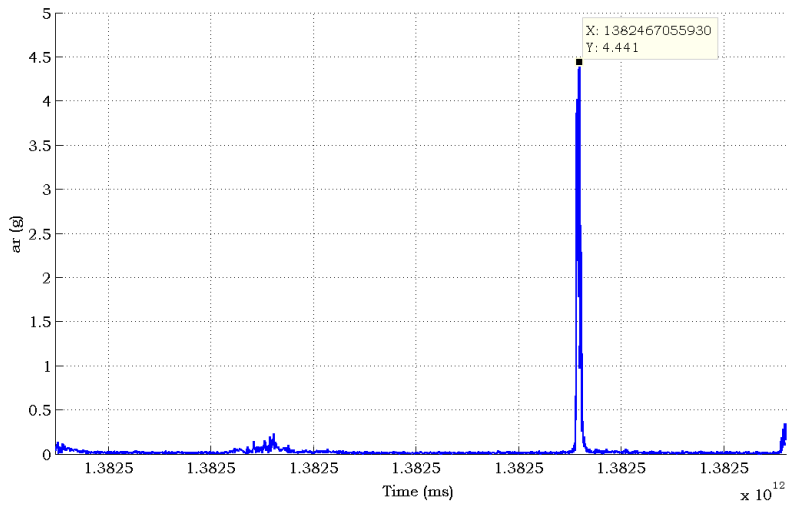


Figure 5.1: a$\vec{r}$ slim pulse evolution.

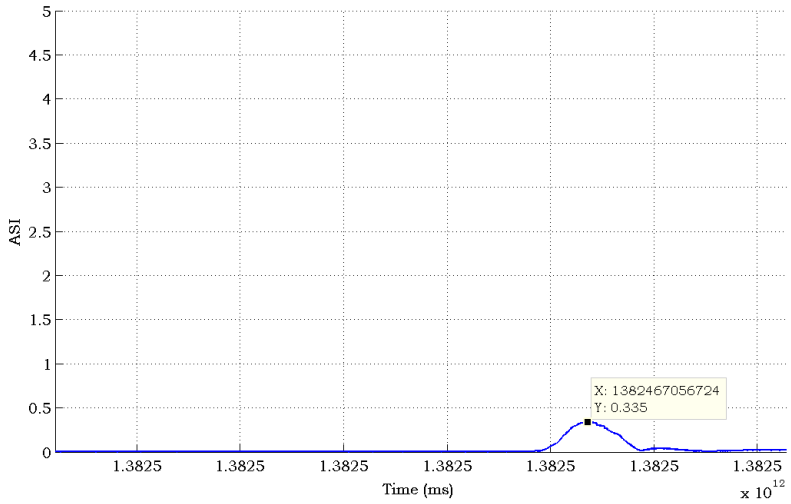


Figure 5.2: ASI value resulting from a$\vec{r}$ slim pulse evolution.

---

test, the maximum ASI value detected was ASI = 0.34, which would lead to an ASI level of A. Figure 5.2 presents the ASI evolution for this test.

This test consisted in violently shaking the device once (one swing). This resulted in a pulse with a small width (slim pulse). Figure 5.3 depicted the amplified view of the pulse. This pulse had the duration of approximately 505 ms. The measurement was made using the closest sample to a$\vec{r}$ = 1g as depicted in figure 5.3. With this method (one swing), it was not possible to achieve significantly higher a$\vec{r}$ and ASI results, therefore it wasn't possible to assess if it is possible to achieve higher ASI levels.

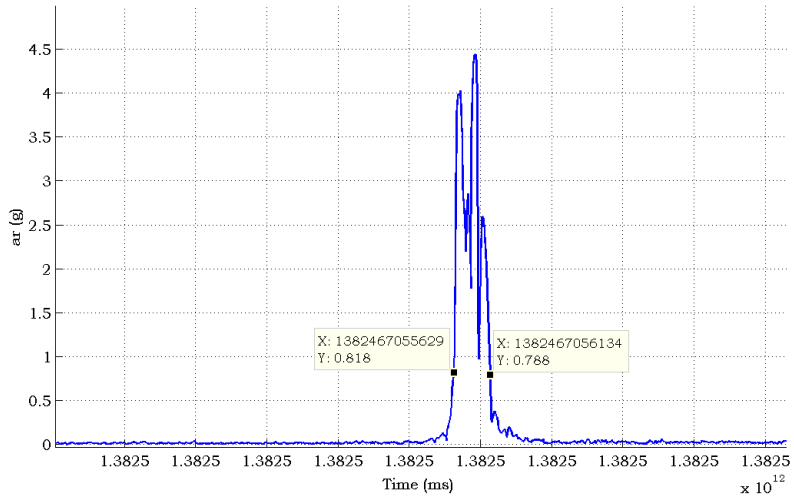


Figure 5.3: a$\vec{r}$ slim pulse amplified view.

As mentioned before with this test it was not possible to assess if the device can detect accelerations that would result in higher ASI levels. The ASI metric is determined using past samples as expression 2.3 presents. This means that for larger a$\vec{r}$ pulse widths, the resulting maximum ASI value obtained should be greater. To prove this statement, another test was performed. It consisted in violently shaking the device several times (several swings) in order to keep the a$\vec{r}$ values high for an greater amount of time. Figure 5.4 presents the results of this test.

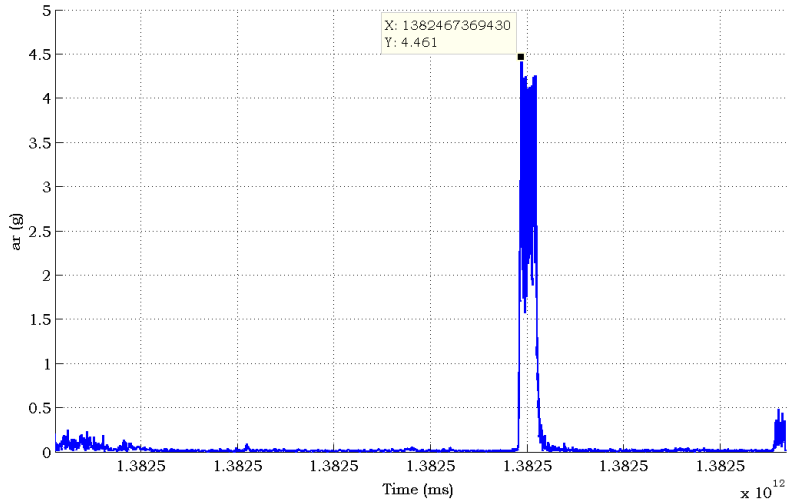


Figure 5.4: a$\vec{r}$ large pulse evolution.

Here the maximum a$\vec{r} = 4.461$g and the pulse width, determined similarly to the previous test, is 1918 ms. The pulse aplified view is depicted in figure 5.6. The maximum ASI value resulting from this pulse is ASI = 1.73, as presented in figure 5.5, which corresponds to an

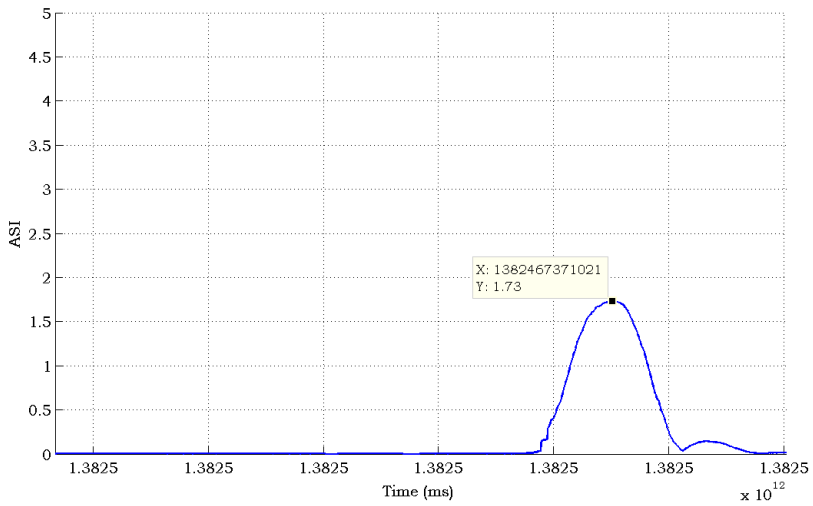


Figure 5.5: ASI value resulting from a$\vec{r}$ large pulse evolution.

ASI level of C. This leads to the conclusion that the application will report an ASI level not only depending on the maximum a$\vec{r}$, but also on the physics of the collision.

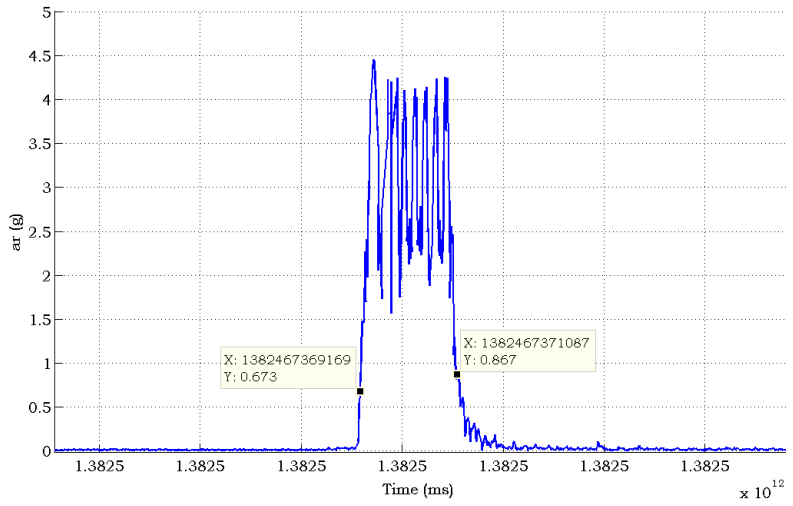


Figure 5.6: a$\vec{r}$ large pulse amplified view.

5.1.2 Rollover simulation and results

To validate a rollover occurrence a minimum of 45 degree change from the device initial position, over its Z axis (azimuth) should occur. The validation of rollover takes into consideration the mean speed of the vehicle and its instant speed after the position change. Here only the position change is tested.

The test performed aimed to verify if the output of the sensor fusion technique was accurate. It consisted into changing the device from portrait mode (initial position) to landscape mode over a flat table. This test enabled a rotation over the device's Z axis only, leaving the X and Y axis rotation null. Figures 5.7, 5.8 and 5.9 present the result of the test.

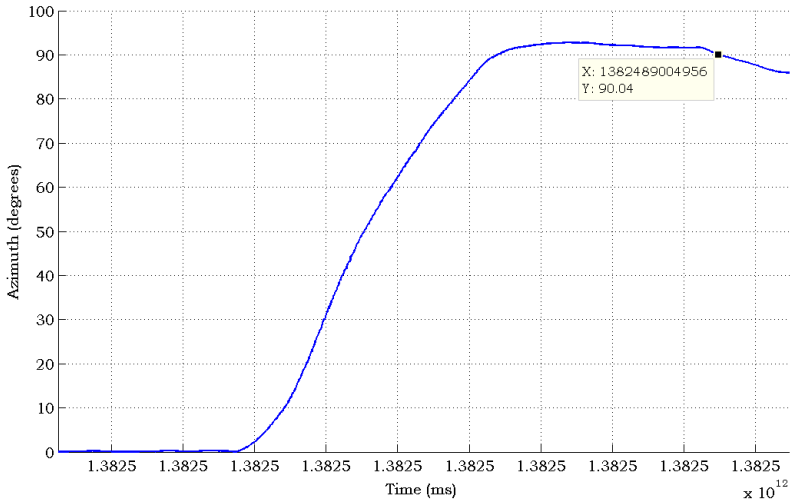


Figure 5.7: Azimuth (Z axis rotation) evolution.

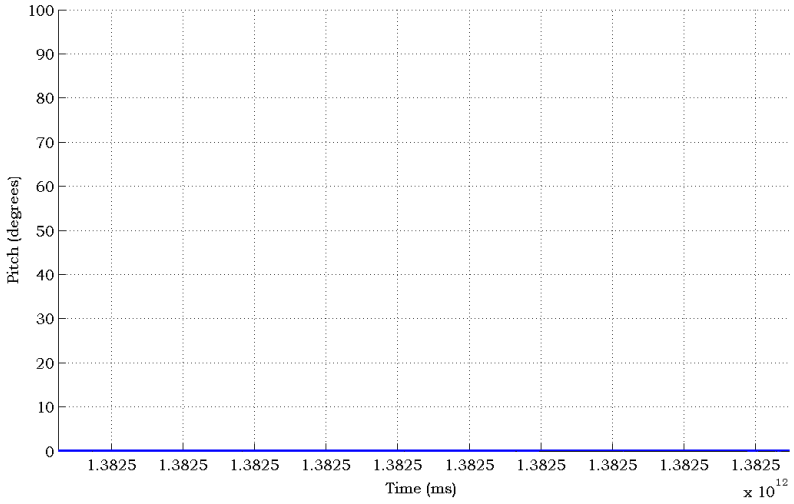


Figure 5.8: Pitch (X axis rotation) evolution.

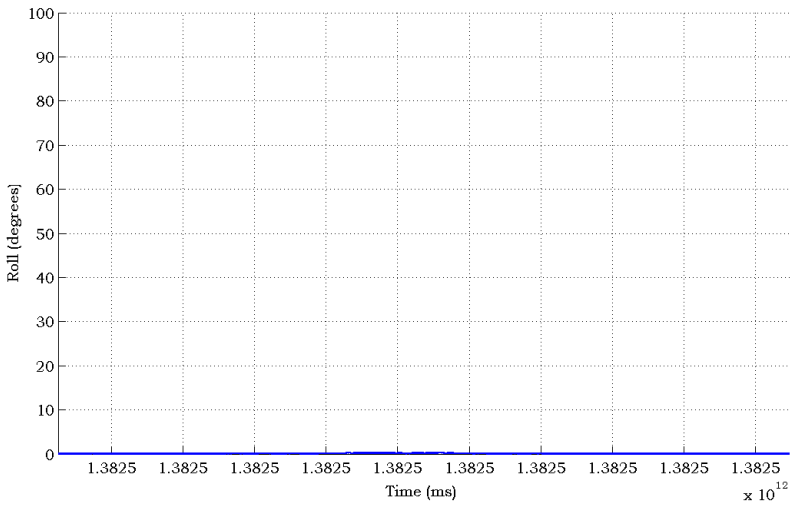


Figure 5.9: Roll (Y axis rotation) evolution.

Its possible to verify in figures 5.7, 5.8 and 5.9 that the sensor fusion outputs approximated values, which are fairly accurate. The position change between portrait and landscape is 90 degrees, which the sensor fusion technique captured correctly. The output only present variations in the Z axis, as expected. With these results, its possible to conclude that the device position is being correctly detected, which allows the application to detect rollovers.

5.2 Robustness tests

After validating the capability to detect accidents based in the device sensors, its important to capture the application behaviour in different situations to assess its robustness, i.e, its ability to respond as expected in different scenarios.

The tests were performed to the application, at runtime, and its behaviour observed and registered in table 5.1, 5.2 and 5.3.

Rotation detected ($\theta \geq 45$)		
Scenario	**Response**	**Elapsed time from detection (ms)**
$\bar{v} = 21$ and $\vec{v}_i = 4$	Nothing happens	-
$\bar{v} = 19$ and $\vec{v}_i = 10$	Nothing happens	-
$\bar{v} = 19$ and $\vec{v}_i = 4$	Nothing happens	-
$\bar{v} = 20$ and $\vec{v}_i = 4$	Countdown activity launch	625

Table 5.1: Rollover detection robustness tests

The first robustness test aimed to verify the rollover validation. It was performed with simulated mean speed ($\bar{v}$) and instant speed ($\vec{v}_i$) values. Once the application detects a rollover, it responds by launching the Countdown activity. Table 5.1 shows that the application responds as expected, i.e, it only validates the rollover occurrence when $\bar{v} \geq 20$ and $\vec{v}_i \leq 5$. The elapsed time between the rollover validation and the launch of the Countdown activity is 690 ms, which shows that the application responds quickly.

When the Countdown activity is launched, if the countdown is not interrupted, HDy Copilot proceeds to request help. Table 5.2 presents the elapsed times of the different help requests for the same test. These times are the result of one test only and are presented only to provide an idea of how the help request is handled.

Start event	**End event**	**Elapsed time (ms)**
Countdown activity launch	Send data to OBU	10494
Countdown activity launch	Send SMS	10094
SMS delivered	Call activity launch	5093

Table 5.2: Help request procedure elapsed times

At this particular test, the countdown time was set for 10 seconds (1000ms) and its possible to verify, in table 5.2, that both the SMSs and data to the OBU, were only sent after the countdown terminated. After the SMSs are sent the call will be only performed after the delivered confirmation is received. This process time depends on the MNO, i.e, does not depend on the application directly and therefore is not accounted for in the test. Once the SMS delivered confirmation is received, the application waits 5 seconds (5000ms) to then

launch the call activity. This waiting time was proven necessary throughout the application development, to guarantee that the GSM connectivity is not obstructed, otherwise the call would fail.

The second test performed was to assess the background execution capabilities to detect accidents and present RHW. Table 5.3 presents the results of this test.

Event	Scenario	Response
Collision detected	USB connected	Help request concluded in full
Collision detected	USB disconnected	Help request concluded. No data sent to OBU
Rollover detected	USB connected	Help request concluded in full
Rollover detected	USB disconnected	Help request concluded. No data sent to OBU
RHW received	USB connected	Received activity brought to foreground

Table 5.3: Background execution robustness tests

As table 5.3 presents the application does not depend fully on the USB connection to detect accidents. If the USB connection is down, HDy Copilot's communication with the IT^{2}S platform is terminated, which means that the OBD-II data will cease to be received and therefore the vehicle airbag deployment signal will not trigger an help request. However, this disconnection with the IT^{2}S does not stop the application execution. In the scenario of a real car accident, the device can be thrown out of the smartphone car holder causing the detachment of USB cable. In this situation, the application continues its execution and is able to detect collisions and rollovers and perform the implemented eCall solution and notify the user previously setted contacts.

When disconnected with the IT^{2}S platform, HDy Copilot's also ceases to be able to receive and report RHWs. The application was designed to transmit and received information to/from the OBU through USB only.

When in background and connected to the OBU the application successfully present RHW to the driver, by dismissing the keyguard, if enabled, powering on the screen and transiting to foreground, displaying the Received activity.

The tests performed prove that the application can detect resulting accelerations higher than 4g, which is the minimum required to validate an accident. They also prove that the ADA detects the device's position variation correctly, allowing the detection of rollovers. The execution of the application is also validated.

Appropriate equipment to test the setup is needed to extend the tests and better quantify and characterize the project.

Having the tests and validation performed, i possible now to conclude the dissertation. The next chapter presents the conclusion.

Chapter 6

Conclusions

The main goal of this dissertation was, on the one hand, to develop an AU to deploy applications that took advantage of HEADWAY capabilities. With this done, HEADWAY's potential could be demonstrated. On the other hand, the goal was to develop a smartphone based eCall system to be also integrated in the AU.

Smartphone capabilities, namely its hardware resources and its mobile oriented OSs, enable the development of applications in several different areas.

To prove the feasibility of the solution, HDy Copilot was developed for Android. Android OS provide APIs that allow the access to almost all its hardware. Android is open-source, therefore it presented few restrictions. One of the restrictions found was the lack of an API that would allow us to use the GSM radios as an in-band modem to perform a true eCall. The solution adopted was using the SMS followed by a voice call.

HEADWAY is a vehicular communication system, that implements the Facilities layer from the ETSI protocol stack, on its layer structure. This means that there are two types of messages exchanged between OBUs, the DENM and the CAM.

To reach the dissertation goals, HDy Copilot was intended to deal with DENM messages, which carry RHWs. The features implemented, the GUI, RHW manual report and ADA, allowed that and still demonstrated, particularly with the ADA, that smartphones can be a valuable asset in ITS.

HDy Copilot was implemented successfully and its overall functionality was proven to work as expected.

6.1 Future work

Several aspects should be improved, in the future, in order to leave the proof of concept stage and approach a commercial version. The main improvements are the following:

- Assess better the values used to validate a rollover, such has the mean speed and instant speed calculating time, by performing more tests.
- Add integration redundancy. HDy Copilot integration with the OBU depends on the USB connection. This dependency should be minimized. Redundancy could be added with Bluetooth connectivity, which both ends support.
- Improve RHW display. Remaining distance to reach the RHW location could be updated in real-time until the location is reached.

- Add voice warnings. With voice warnings the driver wouldn't need to look to the display.
- Validation in controlled environment with appropriate equipment.

Appendixes

Appendix A

Android development reference guides

A.1 Android application structure

Developing for Android, which is a OS mainly for mobile devices, is different from other programming paradigms where applications are launched with a main() method. Applications in Android are composed by the following components:

- Activities - An activity is an application component that provides a screen (View) that the user can interact with. An application usually consists on multiple activities that are bound to each other. In each application there is a main activity, that is the first to be launched when the user opens an application. Due to the size of the Android devices, only one activity can be displayed at a time. As the user navigates through activities the, system manages them through the use of callback methods. These methods structure the application and implement a lifecycle. They can be overridden and allow the developer to use the existing API's and Java programming features to implement functionality to the activity.
- Services - Contrary to an activity, a service does not provide a user interface. They run in the background to perform long-running operations or work for remote processes. Services can be launched by other application components such as activities.
- Content providers - Any application can save data in the file system, an SQLite database or any other persistent location an application can access. Some applications can have data that can be used by other applications. Content providers manages a shared set of application data.
- Broadcast receivers - This component responds to system wide broadcast announcements. These broadcasts can be generated by the system or by other activities (from the same application or not).

A.1.1 Activity lifecycle

Because of the device's characteristics where Android runs, the size of the screen and battery life, only one activity is displayed at a time. This means that the operating system

must be able to manage the activities lifecycle so that data is displayed correctly when the user interacts with the application and energy efficiency is taken into account to prolong the device's battery life.

An application can be composed of one or more activities. An activity can launch another activity, whether it is from the same application or from a different one. To manage this flow of activities, Android OS as a well defined activity lifecycle, depicted in figure A.1.

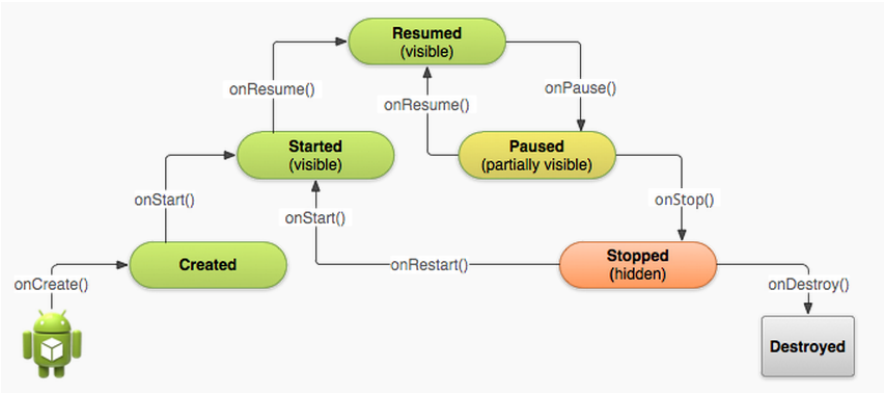


Figure A.1: Android Activity lifecycle *Source: [17]*

Each activity has its lifecycle. The lifecycle of an activity is implemented trough the lifecycle callback methods. The callback methods are evoked when an activity must change its state. An activity has five possible states, created, started, resumed, paused, and stopped. However only three of them can be static, meaning that the activity can exist in one of these states for an extended period of time. The static states are the following:

- Resumed - In this state the activity is in foreground and the user can interact with it.
- Paused - In this state the activity does not receive user inputs and cannot execute code. Is partially obscured by another activity.
- Stopped - In this state the activity is hidden and not visible to the user, i.e, its in background. No code is executed. The activity instance and is variables are retained.

The other states (created and started) are transient and the system quickly moves through them, by calling the next lifecycle callback method. Each call back method can be overridden so that developers can control the application. According with the Android developers website [17] the lifecycle callbacks are the following:

- onCreate() - Called when the activity is first created. Here the developer can create views (layout elements), bind data to lists, instantiate/initialize objects and refers variables to it, etc.
- onStart() - Called just before the activity become visible. The developer can initialize and initiate data.
- onResume() - Called just before the activity start interacting with the user. It can be user to initialize data or to reinitialize it if the activity is coming from the stopped state. The developer can reinitialise the sensors or location readings, etc.

- onPaused() - Called when the system is about to resume another activity. Here the developer can save data, free CPU resources such as sensor or location readings. It should be completed quickly, because other activities will not resume before this method returns.
- onStop() - Called when the activity is no longer visible. The developer can use this to free CPU resources.
- onRestart() - Called after the activity has been stopped, just prior to it being started again.
- onDestroy() - Called just before the activity is destroyed.

Appendix B

Android design guidelines (version 4.0.3 or higher)

Every operating system as its own design language. The design language is usually used to hint functionalities and to create a working mode throughout all the operating system.

Android has its own design language and application developers should follow it, in order to provide a familiar and consistent experience to users.

B.1 Gestures

This OS was mainly developed for touch interaction, i.e, users interact with the system by touching the device's screen. The system allows for several touch gestures to reproduce different interaction results. The gestures supported are the following:

- Touch/ Tap - Recognized when the action press and lift happen quickly. Triggers the default functionality for a given item.
- Long Press - Recognized when the action press, wait and lift happens. Enters data selection mode. Allows to select an item in a view an act upon data using a contextual action bar.
- Swipe - Recognized when the action press, move and lift occur. Scrolls through content and can be also be used to navigate.
- Drag - Recognized when the action long press, move and lift occur. Rearranges data or within a view or moves it into a container (e.g drag file to folder)
- Double touch - Recognized when two touches in quick succession occur. Zooms into content. Also used as secondary gesture for text selection.
- Pinch open - Recognized when the action two fingers press, move outwards and lift occurs. Zooms into content.
- Pinch close - Recognized when the action two fingers press, move inwards and lift occurs. Zooms out of content.

B.2 Android navigation

Android UI basic aspect is composed by a Home screen, an all application screen and a recent screen. The Home screen is a customizable space that can contain application shortcuts, folders and widgets. There can be several home screens and navigation between them is done through the swipe gesture. Commonly present through all home screens is the favourites tray, that can host some of our favourites shortcuts or folders and also host the all application screen button. The all application screen displays all the applications and widgets installed on the device. The recent screen displays all the recent launched applications.

The most recent Android devices possess three navigation buttons. A back button, to navigate back in time, i.e, close the present activity and open the previous one, the home button, to stop activities and display the home screen and the recent button, that displays the recent screen. Some devices instead of the recent button have a menu button.

A status bar is always accessible throughout the applications. This status bar displays application notifications, date and time, battery level indicator and network signal strength indicator. The user can swipe the bar down for it to display additional info.

The design language of Android is also applied through its applications. Users expect that applications behave in a similar ways. Still developers have a number of choices available to design the application. Applications that only have one Activity have different design requirements from applications that have multiple activities. The common application UI is depicted in figure B.1.

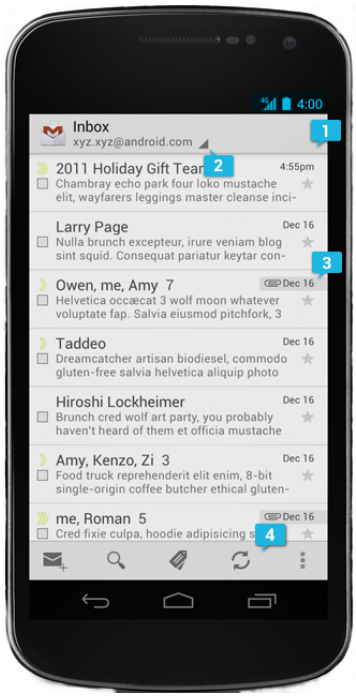


Figure B.1: Common Android application UI. *Source: [17].*

B.3 Application design structure

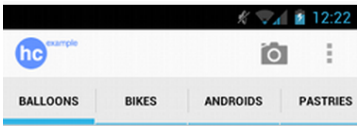
The application UI generally is composed by a main action bar (1), view control (2), content area (3) and a split action bar (4). The main action bar is where its implemented the command and control center of the application. It includes the navigation elements for application navigation (e.g. change activity) as well as the most important application actions. The view control allows users to switch between different views (different arrangements of data or different functional aspects of the application). The content area is the area where the content of the activity is presented. The split action bar provides a way to distribute important actions, that don't fit in the main action bar, into different bars. These can be located bellow the main action bar or bellow the content area.

B.4 Application navigation

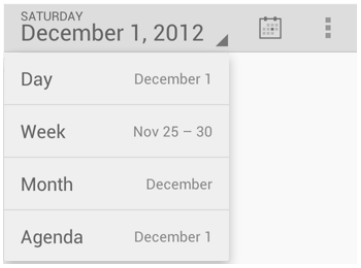
An important aspect of the application design is its navigation scheme. The navigation between activities can be implemented in the action bar with tabs or a drop down menu, or even through the use of navigation drawers. Figure B.2 depicts the user interface of these navigation schemes.

Each of these types of navigation should be implemented in different circumstances. The recommendation in [17] are the following:

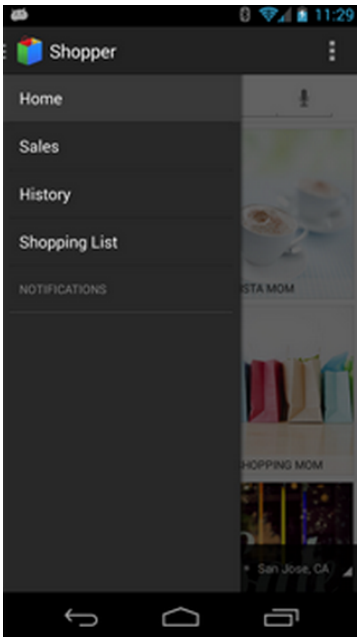
- Tabs
  - If the user is expected to change views frequently
  - If there is a limit of three tabs
  - If the user should be highly aware of the other views
- Spinner
  - If the user is changing between views of the same data set (e.g. calendar events viewed by day, week or month) or data sets of the same views (e.g. content of different accounts).
  - If screen space used by tabs is not affordable.
- Navigation drawer
  - To provide deep navigation branches.
  - To provide quick navigation to views which don't have direct relationships between each other.
  - To save screen space.
  - If there are a large number of switchable views.



(a) Navigation tabs. *Source: [17]*



(b) Drop down spinner. *Source: [17]*



(c) Navigation drawer. *Source: [17]*

Figure B.2: Action bar navigation types.

Bibliography

[1] European Commission. Road fatalities in the EU since 2001. [Available online at] http://ec.europa.eu/transport/road_safety/specialist/statistics/, [Accessed] 4 December 2012.

[2] European Telecommunications Standards Institute (ETSI). [Available online at] http://www.etsi.org/, [Accessed] 22 July 2013.

[3] Institute of Electrical and Electronics Engineers (IEEE). [Available online at] http://www.ieee.org/index.html, [Accessed] 22 July 2013.

[4] Brisa Innovation. HEADWAY - Connecting vehicles and highways. [Available online at] http://www.brisainovacao.pt/en/innovation/projects/headway, [Accessed] 12 October 2012.

[5] Figueiredo L. *Sistemas Inteligentes de Transporte*. PhD thesis, Faculdade de Engenharia da Universidade do Porto, February 2005.

[6] Osório L. Fortes F. Serrador A., Prata R. Vem aí a Via Verde de Segunda Geração. *Politecnia*, May 2007.

[7] Yan Zhang Hassna Moustafa. *Vehicular Networks, Techniques, Standards and Applications*. Auerbach Publications, 2009.

[8] Antonella Molinaro Claudia Campolo. Multichannel Communications in Vehicular Ad Hoc Networks: A Survey. *IEEE Communications Magazine*, 2013.

[9] Meireles T. Ferreira N. Mar P. Fonseca J. Carona D. Serrador A. Lopes J. Matos J., Oliveira A. Emergent Vehicular Communications: Applications, Standards and Implementation. *Proc URSI Seminar of the Portuguese Committee*, 1:1–1, September 2010.

[10] Matos J. N. Oliveira A. S. R. IT2S Board Description. March 2013.

[11] Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service. Technical report, ETSI, March 2011.

[12] Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service. Technical report, ETSI, September 2010.

[13] IDC. Nearly 1 billion smart connected devices shipped in 2011 with shipments expected to double by 2016. [Available online at] http://www.idc.com/getdoc.jsp?containerId=prUS23398412, [Accessed] 28 October 2012.

[14] Gartner. Asia/Pacific Led Worldwide Mobile Phone Sales to Growth in First Quarter of 2013. [Available online at] http://www.gartner.com/newsroom/id/2482816, [Accessed] 14 October 2012.

[15] Google. About the Android Open Source Project. [Available online at] http://source.android.com/about/index.html, [Accessed] 13 October 2012.

[16] Google. Android Main Website. [Available online at] http://www.android.com/, [Accessed] 12 November 2012.

[17] Google. Android Developers Website. [Available online at] http://developer.android.com/index.html, [Accessed] 13 November 2013.

[18] Google. Google Play Developers Distribuition Contract. [Available online at] https://play.google.com/about/developer-distribution-agreement.html, [Accessed] 13 November 2012.

[19] Reto Meier. *Professional Android^{TM} 4 Application Development*. John Wiley & Sons, Inc., Indianapolis, Indiana, 2012.

[20] Berkeley Software Distribution (BSD). [Available online at] http://www.bsd.org/, [Accessed] 11 January 2013.

[21] Apple. Apple Developer Website. [Available online at] https://developer.apple.com/technologies/ios/, [Accessed] 30 October 2012.

[22] Apple. Enrolling in Apple Developer Programs. [Available online at] https://developer.apple.com/programs/start/standard/, [Accessed] 05 June 2013.

[23] Apple. MFi Program. [Available online at] https://developer.apple.com/programs/mfi/, [Accessed] 05 June 2013.

[24] PhoneGap Website. [Available online at] http://phonegap.com/, [Accessed] 23 April 2013.

[25] Georgijs Kanonirs Reinholds Zviedris Leo Selavo Girts Starzdins, Artis Mednis. Towards Vehicular Sensor Networks with Android Smartphones for Road Surface Monitoring. *The Second International Workshop on Networks of Cooperating Objects (CONET'11), Electronic Proceedings of CPSWeek'11*, 2011.

[26] Juan Carlos Cano Pietro Manzoni Jorge Zaldivar, Carlos T. Calafate. Providing Accident Detection in Vehicular Networks Through OBD-II Devices and Android-based Smartphones. *5th IEEE Workshop On User Mobility and Vehicular Networks*, 2011.

[27] Andrew Austin. Eco Driving Tutor Application for Android. Technical report, Lancaster University.

[28] DRIVE-IN | CMU Portugal. [Available online at] http://drive-in.cmuportugal.org/, [Accessed] 16 October 2012.

[29] Scott J. Weiner. Feasibility of a 802.11 vanet based car accident alert system. 2010.

[30] Brian Dougherty Adam Albright Chris Thompson, Jules White and Douglas C. Schmidt. Using smartphones to detect car accidents and provide situational awareness to emergency responders. Technical report, Institute for Software Integrated Systems.

[31] Vartika Mehta Deepak Kumar, Deepak Punetha. Design and Realization of the Accelerometer based Transportation System. *International Journal of Computer Applications*, July 2012.

[32] European Committee for Standardization. Road restraint systems - part 1: Terminology and general criteria for test methods. Technical report, July 2010.

[33] European Committee for Standardization. Road restraint systems - part 2: Performance classes, impact test acceptance criteria and test methods for safety barriers including vehicle parapets. Technical report, July 2010.

[34] Hampton C. Gabler Douglas Gabauer. Evaluation of the Acceleration Severity Index Threshold Values Utilizing Event Data Recorder Technology. Technical report, College of Engineering Rowan University, 2003.

[35] Dr. M. Shojaati. Correlation between injury risk and impact severity index ASI. *3rd SwissTransport Research Conference*, 2003.

[36] Anders Eriksson ElsMarie Henriksson, Mats Ostrom. Preventability of Vehicle-Related Fatalities. April 2000.

[37] European Commission. eCall: automated emergency call for road accidents mandatory in cars from 2015. [Avaliable online at] http://europa.eu/rapid/press-release_IP-13-534_en.htm, 13 June 2013.

[38] iCar Support. eCall. [Available online at] http://www.icarsupport.eu/ecall/, [Accessed] 19 March 2013.

[39] 3rd Generation Partnership Project. TR 26.967, eCall Data Transfer. Technical report, ETSI, GSMA, 2012.

[40] eCall Driving Group. Recommendations of the DG eCall for the introduction of the pan-European eCall. Technical report, 2006.

[41] UML® Resource Page. [Available online at] http://www.uml.org/, [Accessed] 23 June 2013.

[42] Marting Fowler. *UML Distilled*. Addison-Wesley, 3 edition, September 2003.

[43] Wolfram MathWorld. Harvesine Formula. [Available online at] http://mathworld.wolfram.com/Haversine.html, [Accessed] 13 June 2013.

[44] Thousand Thoughts WordPress Blog. Android Sensor Fusion. [Available online at] http://www.thousand-thoughts.com/2012/03/android-sensor-fusion-tutorial/, [Accessed] 20 February 2013.

[45] MIT License. [Available online at] http://opensource.org/licenses/mit-license.php, [Accessed] 27 May 2013.