TECHNOLOGICAL EDUCATIONAL INSTITUTE OF CRETE

MSc in Informatics & Multimedia

Master's Thesis

Smartphone-Based Telematics for Usage Based Insurance

Prokopios Vavouranakis

Supervisor:
Dr. Spyros Panagiotakis, Assistant Professor

Heraklion, February 2016

Preface

This master thesis has been conducted in Technological Educational Institute of Crete under the supervision of Assistant Professor Spyros Panagiotakis. Object of this thesis is the study and the implementation of a smartphone-based telematics system for usage based insurance (UBI).

At this point I would like to thank my supervisor Mr. Spyros Panagiotakis for the assignment of the diploma thesis and for his decisive contribution in the implementation of this thesis.

Finally I would like to say a big thank you to my parents, Michael and Athena, and my sister Mara, for the unconditional love and support they have given me over the years. There are no words to describe my love and gratitude.

Abstract

This thesis aims to study and implement a smartphone-based telematics system for a usage-based insurance (UBI). The smartphone has been identified as an enabler for future UBI, replacing the vehicle (after-) mounted dedicated hardware with a ubiquitous device with a plurality of sensors, means for data processing and wireless communication.

We implemented and developed an end-to-end system including a telematics android-based application for client's smartphones and a portal to collect, analyze and record driving patterns and score drivers. Driver Behavior monitoring has evolved tremendously in recent years. Monitoring driver behavior, recording their driving events (safe and aggressive) and giving feedback of recorded events can enhance driver safety.

We implemented and developed an android-based application, which can estimate driving behavior using two methods. Using data only from the accelerometer sensor or using orientation data of a sensor fusion method, which combine data from the accelerometer, the gyroscope and the magnetometer.

We can recognize events like hard acceleration, safe acceleration, sharp left turn, safe right turn, sharp lane left lane change etc. The application to improve the driving behavior of the driver, displays some hint messages, after each bad-driving event. For every trip, the app calculates a score for the driving behavior of the driver and gives him a rating to help him improving his behavior in the next trips. All trips data such as bad driving events, which take place during the trip, are stored in a database and the driver has the opportunity to review and analyze them whenever he wants.

Also, we developed a portal, where we can have access to an overall dashboard of all registered drivers. In the portal are presented scores, behaviors, trips reports and routes in maps of all recorded trips. With this way, we help the insurance carriers to assess better the risk of the drivers.

Table of Contents

List of Figures

List of Tables

Chapter 1 – Introduction

1.1 Basic Concept

Usage-based insurance (UBI) [1] is a type of vehicle insurance whereby the costs are dependent upon type of vehicle used, measured against time, distance, behavior and place.

This differs from traditional insurance, which attempts to differentiate and reward "safe" drivers, giving them lower premiums and/or a no-claims bonus. However, conventional differentiation is a reflection of history rather than present patterns of behavior. This means that it may take a long time before safer (or more reckless) patterns of driving and changes in lifestyle feed through into premiums.
The simplest form of usage-based insurance bases the insurance costs simply on the number of miles driven. However, the general concept of pay as you drive includes any scheme where the insurance costs may depend not just on how much you drive but how, where, and when one drives. [2]

Pay as you drive (PAYD) means that the insurance premium is calculated dynamically, typically according to the amount driven. There are three types of usage-based insurance:

- Coverage is based on the odometer reading of the vehicle.
- Coverage is based on mileage aggregated from GPS data, or the number of minutes the vehicle is being used as recorded by a vehicle-independent module transmitting data via cellphone or RF technology. [3]
- Coverage is based on other data collected from the vehicle, including speed and time-of-day information, historic riskiness of the road, driving actions in addition to distance or time travelled.

The formula can be a simple function of the number of miles driven, or can vary according to the type of driving or the identity of the driver. Once the basic scheme is in place, it is possible to add further details, such as an extra risk premium if someone drives too long without a break, uses their mobile phone while driving, or travels at an excessive speed. Telematics usage-based insurance provides a much more immediate feedback loop to the driver, [1] by changing the cost of insurance dynamically with a change of risk. This means drivers have a stronger incentive to adopt safer practices. For example, if a commuter switches to public transport or to working at home, this immediately reduces the risk of rush hour accidents. With usage-based insurance, this reduction would be immediately reflected in the cost of car insurance for that month.

So the proposal for the user is simple. The user has to download an application in his smartphone, open it when he is driving and let his insurance company to monitor his driving behavior. Where his vehicle is driven, how fast he drives, how hard he breaks, how hard he corners and so on. In return the insurance will give him a discount on insurance premiums.

This may seem disturbing, but it should not be a surprise. If someone considers that Internet users already gladly offer their data against free services, the only

surprise is that the insurance based on the recording data is not widespread. The Progressive Insurance [4], the largest insurance company using such methods in the USA, found after analysis of billion of miles and relevant data that key points in driving behavior such as actual miles traveled, braking and time of the day give more than the twice predictabilities compared to tradition variables such as age of the driver, gender, the manufacturer and the model of the insured vehicle. The average discount on insurance premiums for a driver who agrees to record his driving behavior amounts to 10-15%.

In the future, anyone who does not agree to record his driving behavior may not be required to pay higher insurance premiums, but most companies will not even accept to insure his car. Insurance based on telematics can find great appeal, as no longer required special devices installed in the car but it is enough a simple download of an application in your smartphone. In addition, smartphones are made especially for communication. On the other hand special devices need some kind of transmitter.

Insurance programs based on telematics [2] would mean big changes for road safety. The insurance application on the mobile phone will notify you when you brake too abruptly or run too much, and tame the way you drive. The fact is that people drive more carefully, simply because they know that their driving behavior is recorded. The more expensive insurance premiums act as penalty for recklessness driving behavior.

1.2 Purpose and Structure of Thesis

Purpose of thesis is to implement a usage based auto-insurance information system, which consists of two elements. The first element is an android-based application for smartphones and tablets, which detects the driver's behavior by analyzing the collected data from device's sensors. For this reason we study various driving detection methods using smartphone's sensor in order to find which method or methods are the best for our implementation. The second element is an e-platform, where someone can have access to all data (trip's information, routes of trips, graphs of sensor's data) of all drivers, of the insurance company.

In chapter 2 is presented the concept of usage-based insurance, the definition and the advantages of UBI. Also we talked about the challenges and future of UBI. In chapter 3 are presented the various types of sensors, which are in-built in smartphones and tablets. Also we present a survey about driver classification and driving behavior recognition using a smartphone's sensors. In chapter 4 is presented the vehicle coordinate system and how we calibrate the device (re-orient the sensor's axis) in order to get always one type of values, regardless the position of the device inside the car. In chapter 5 we present the sensor fusion method, which used to detect the driving behavior. This method uses the accelerometer, geomagnetic field sensor and the gyroscope in order to get the orientation of the vehicle.
In chapter 6, are presented the selected two driver detection methods. The first method is to evaluate the data we get from the accelerometer sensor and in the second method we evaluate the orientation data we get from the sensor fusion

method. We present thresholds and graphs for various driving events for both methods and we analyze the driver behavior detection algorithm.

In chapter 7 we discuss about the android-based application, which detects the driver's behavior. The features of the application are presented and the way we use the application. Also is presented the e-platform of the UBI information system, how we use it and what data are collected and presented.

In chapter 8 are presented the technologies and the tools, which we used to implement the android-based application and the e-platform. In the end, Chapter 9 summarizes the findings of thesis and presents the potential benefits of using a UBI system.

Chapter 2 – Usage Based Insurance (UBI)

2.1 Definition of UBI

Usage-based insurance (UBI) [2] also known as pay as you drive (PAYD) and pay how you drive (PHYD) and mile-based auto insurance is a type of vehicle insurance whereby the costs are dependent upon type of vehicle used, measured against time, distance, behavior and place.

Usage-Based Insurance is a recent innovation by auto insurers that more closely aligns driving behaviors with premium rates for auto insurance. Mileage and driving behaviors are tracked using odometer readings or in-vehicle telecommunication devices (telematics) that are usually self-installed into a special vehicle port or already integrated in original equipment installed by car manufactures. The basic idea of telematics auto insurance is that a driver's behavior is monitored directly while the person drives. These telematics devices measure a number of elements of interest to underwriters: miles driven; time of day; where the vehicle is driven (GPS); rapid acceleration; hard breaking; hard cornering; and air bag deployment. The level of data collected generally reflects the telematics technology employed and the policyholders' willingness to share personal data.

The insurance company then assesses the data and charges insurance premiums accordingly. For example, a driver who drives long distance at high speed will be charged a higher rate than a driver who drives short distances at slower speeds. With UBI, premiums are collected using a variety of methods, including utilizing the gas pump, debit accounts, direct billing and smart card systems.

The first UBI programs began to surface in the U.S. about a decade ago, when Progressive Insurance Company and General Motors Assurance Company (GMAC) began to offer mileage-linked discounts through combined GPS technology and cellular systems that tracked miles driven. These discounts were (and still are) often combined with ancillary benefits like roadside assistance and vehicle theft recovery. Recent accelerations in technology have increased the effectiveness and cost of using telematics, enabling insurers to capture not just how many miles people drive, but how and when they drive too. The result has been the growth of several UBI variations, including Pay-As-You-Drive (PAYD), Pay-How-You-Drive (PHYD), Pay-As-You-Go, and Distance-Based Insurance.

2.2 Pricing of UBI

The pricing scheme for UBI deviates [2] greatly from that of traditional auto insurance. Traditional auto insurance relies on actuarial studies of aggregated historical data to produce rating factors that include driving record, credit-based insurance score, personal characteristics (age, gender, and marital status), vehicle type, living location, vehicle use, previous claims, liability limits, and deductibles. Premium discounts on traditional auto insurance is usually limited to the bundling of insurance on multiple vehicles or types of insurance, insurance with

the same carrier, protection devices (like airbags), driving courses and home to work mileage.

Policyholders tend to think of traditional auto insurance as a fixed cost, assessed annually and usually paid for in lump sums on an annual, semi-annual, or quarterly basis. However, studies show that there is a strong correlation between claim and loss costs and mileage driven, particularly within existing price rating factors (such as class and territory). For this reason, many UBI programs seek to convert the fixed costs associated with mileage driven into variable costs that can be used in conjunction with other rating factors in the premium calculation. UBI has the advantage of utilizing individual and current driving behaviors, rather than relying on aggregated statistics and driving records that are based on past trends and events, making premium pricing more individualized and precise.

2.3 Advantages of UBI

UBI programs offer many advantages [2] to insurers, consumers and society. Linking insurance premiums more closely to actual individual vehicle or fleet performance allows insurers to more accurately price premiums. This increases affordability for lower-risk drivers, many of whom are also lower-income drivers. It also gives consumers the ability to control their premium costs by incenting them to reduce miles driven and adopt safer driving habits. Fewer miles and safer driving also aid in reducing accidents, congestion and vehicle emissions, which benefits society.

The use of telematics helps insurers more accurately estimate accident damages and reduce fraud by enabling them to analyze the driving data (such as hard breaking, speed, and time) during an accident. This additional data can also be used by insurers to refine or differentiate UBI products. Additionally, the ancillary safety benefits offered in conjunction with many telematics-based UBI programs also help to lower accident and vehicle theft related costs by improving accident response time, allowing for stolen vehicles to be tracked and recovered, and monitoring driver safety. Telematics also allow fleets to determine the most efficient routes, saving them costs related to personnel, gas and maintenance.

2.4 Challenges

The practice of tracking mileage and behavior information in UBI programs has raised privacy concerns. As a result, some states have enacted legislation requiring disclosure of tracking practices and devices. Additionally, some insurers limit the data they collect. Although not for everyone, acceptance of information sharing is growing as more mainstream technology devices (such as smartphones, tablets, and GPS devices) and social media networks (such as Facebook and MySpace) enter the market.

Implementing a UBI program, particularly one that utilizes telematics, can be costly and resource intensive to the insurer. UBI programs rely heavily on costly technology to capture and sensitize driving data. Additionally, UBI is an emerging area and thus there is still much uncertainty surrounding the selection and

interpretation of driving data and how that data should be integrated into existing or new price structures to maintain profitability. This is particularly important, as the transitioning of lower-risk drivers into UBI programs that offer lower premium could put pressure on overall insurer profitability.

Insurers must also manage regulatory requirements within the states that they do business. Many states require insures to obtain approval for the use of new rating plans. Rate filings usually must include statistical data that supports the proposed new rating structure. Although there are general studies demonstrating the link between mileage and risk, individual driving data and UBI plan specifics are considered proprietary information of the insurer. This can make it difficult for an insurer who does not have past UBI experience. Other requirements that could prevent certain UBI programs include the need for continuous insurance coverage, upfront statement of premium charge, set expiration date, and guaranteed renewability. However, it should be noted that a Georgia Institute of Technology survey of state insurance regulations (2002) found that the majority of states had no regulatory restrictions that would prevent PAYD programs from being implemented.

2.4 Implementations in USA

Metromile. Metromile is a California-based insurance startup funded by New Enterprise Associates, Index Ventures, National General Insurance/Amtrust Financial, and other investors. It offers a driving app and a pay-per-mile insurance product using a device that connects to the OBD-II port of all automobiles built after 1996. Metromile does not use behavioral statistics like type of driving or time of day to price their insurance. They offer consumers a fixed base rate per month plus a per-mile-rate ranging from 2 to 11 cents per mile, taking into account all traditional insurance risk factors. Drivers who drive less than the average (10,000 miles a year) will tend to save. Metromile allows users to opt out of GPS tracking, never sells consumer data to 3rd parties, and does not penalize consumers for behavioral driving habits. Metromile is currently licensed to sell auto insurance in California, Oregon, Washington, Virginia and Illinois (as of July 2015). [49] More states are expected to roll out shortly.

Progressive. Snapshot is a car insurance program developed by Progressive Insurance in the United States. [50] It is a voluntary, behavior-based insurance program that gives drivers a customized insurance rate based on how, how much, and when their car is driven. Snapshot is currently available in 46 states plus the District of Columbia. Because insurance is regulated at the state level, Snapshot is currently not available in Alaska, California, Hawaii, and North Carolina.

Driving data is transmitted to the company using an on-board telematic device. The device connects to a car's Onboard Diagnostic (OBD-II) port (all automobiles built after 1996 have an OBD-II.) and transmits speed, time of day and number of miles the car is driven. Cars that are driven less often, in less risky ways and at less risky times of day can receive large discounts. Progressive has received patents on its methods and systems of implementing usage-based insurance and

has licensed these methods and systems to other companies. Progressive has service marks pending on the terms Pay As You Drive and Pay How You Drive.

Allstate. Allstate announced on October 8, 2012 that it has expanded its usage-based auto insurance product, Drive Wise, to four additional states including New York and New Jersey. [51] As of October 2012 Drive Wise is currently available in: Colorado, Michigan, New Jersey, New York, Arizona, Illinois, and Ohio. Allstate's usage-based insurance product, Drive Wise, gets installed into a car's onboard diagnostic port, near the steering column in most cars. Allstate said its usage-based insurance measures things such as mileage, braking, speed, and time of day when a customer is driving. Using that data, Allstate calculates a driving discount for each customer using its telematics technology.

One of the big advantages with Drive Wise is that it can constantly provide feedback to the consumers for as long as they keep the device in the car. Allstate's Drive Wise utilizes data from a monitoring device plugged into a car's onboard diagnostic port. Of the drivers earning a discount, the average savings is nearly 14 percent per vehicle. More than 10 percent of all new Allstate customers are opting to participate in this coverage.

Liberty Mutual Insurance. Onboard Advisor is a commercial lines pay-how-you-drive, PHYD, or "safety-driven" insurance product by Liberty Mutual Agency Corporation. It offers up to 40% discount to commercial and private fleets based on how safely they actually drive.

National General Insurance. National General Insurance is one of the first and largest auto insurance companies to institute a Pay-As-You-Drive (PAYD) program in the United States back in 2004. [52] The National General Insurance Low-Mileage Discount is an innovative program offered to OnStar subscribers in 34 states, where those who drive less pay less on their auto insurance. This opt-in program is the first of its kind [53] leveraging state-of-the-art technology using OnStar to allow customers who drive fewer miles to benefit from substantial savings. Eligible active OnStar subscribers sign up to save on their premiums if they drive less than 15,000 miles annually. Subscribers who drive even less than that can save even more (up to 54%).

Under the program, new National General Insurance customers receive an automatic insurance discount of approximately 26 percent upon [54] enrollment (existing OnStar customers receive a discount based on historical mileage). With the subscriber's permission, the odometer reading from his or her monthly OnStar Vehicle Diagnostics report is forwarded to National General Insurance. Based on those readings, the company will decrease the premium using discount tiers corresponding to miles driven.

Information sent from OnStar to National General Insurance pertains solely to mileage, and no additional data is gathered or used for any purpose other than to help manage transportation costs. Customers who drive more than 15,000 miles per year are not penalized and all OnStar customers receive an insurance discount simply for having an active OnStar subscription.

2.5 Future of UBI

UBI is poised for rapid growth in the U.S. According to SMA Research [2], approximately 36 percent of all auto insurance carriers are expected to use telematics UBI by 2020. Based on a May 2014 CIPR survey of 47 U.S. state and territory insurance departments, in all but five jurisdictions – California, New Mexico, Puerto Rico, Virgin Islands, and Guam - insurers currently offer telematics UBI policies. In twenty-three states, there are more than five insurance companies active in the telematics UBI market. The CIPR survey is part of a recently released ***CIPR study, Usage-Based Insurance and Vehicle Telematics: Insurance Market and Regulatory Implications***, on how technological advances in telematics are driving changes in the insurance market and its impact on insurers.

Telematics-based UBI growth is being propelled by technology advances, which continue to substantially improve the cost, convenience, and effectiveness of using telematics devices. It is through the use of telematics that insurers are able to collect driving data that better enable them to more closely link a driver's individual risk with premium. Through UBI programs, insurers are able to differentiate products, gain competitive advantage, and attract low-risk policyholders. Recognition of the societal benefits and growing consumer acceptance of personal data collection will only serve to further increase demand for telematics-based UBI products in the future.

Chapter 3 – Detecting Driver Behavior using Smartphone Sensors

3.1 Smartphone Hardware Sensors

Sensors made the smartphones smart. Sensor is a converter that measures a physical quantity and converts it into a signal, which can be read by an observer or by an instrument. There are various types of sensors, which are currently being used in analyzing driver behavior. These sensors are:

a) **Accelerometer.** An accelerometer [5] is an electromechanical device that will measure acceleration forces. These forces may be static (z axis), like the constant force of gravity pulling at your feet, or they could be dynamic (x, y axis) caused by moving or vibrating the accelerometer.
b) **GPS.** GPS [6] is a satellite based Navigation tracking, often with a map showing where you have been. It gives us the value of longitude and latitude, which determines the point of location on earth.
c) **Gyroscope.** Gyroscope [7] detects the current orientation of the device, or changes in the orientation of the device. Orientation can be computed from the angular rate that is detected by the gyroscope. It basically works on the principle of angular momentum and it is expressed in rad/s on 3-axis.
d) **Camera.**
e) **Microphone.**
f) **Magnetometer.** Magnetometers [8] are measurements instruments used for two general purposes-to measure the magnetization of a magnetic material like a Ferro magnet, or to measure the magnetic strength and the direction of the magnetic field at a point in space.

3.2 Methods of Detecting Driver Behavior

There are many researchers who have tried to detect driving behavior using mobile phone sensors.

Singh [9] et al. developed an android application, which first collects data from accelerometer sensor and GPS sensor. Also collects data from the microphone of the smartphone. Then analyzes the data and detects rash driving events (speed breaker, left turn, right turn, left lane change, right lane change, sudden braking, sudden acceleration). Rash driving events are verified using "Ground Truth". Also the data from the accelerometer are combined with the data from the microphone to detect more rash driving events (lane change is not accompanied with indicator sound) or traffic (slow speed with frequent honking).

Fazeen [10] et al. developed an android application for smartphones for detecting driver's behavior. They have used the accelerometer sensor of smartphones (which are integrated inside the cars) to collect and detect various driver styles or road conditions. They have analyzed the data from the accelerometer sensor (x axis, y axis) to measure the driver's direct control of the vehicle as they steer, accelerate or braking. We can detect the difference between safe and sudden acceleration/declaration because safe acceleration/declaration never reach a g-force of more than +0.3/-0.3 g. On the other hand sudden acceleration/declaration reach +0.5/-0.5 g. Safe left/right lane change never

reaches a g-force of less than -0.1g/+0.1g and sudden or unsafe lane change (left/right) reaches a g-force over -0.5g/+0.5g. The results of the experiment showed us that the best phone placement location inside the car is the center console of the car. This location gave us the best relative data with the lowest engine feedback and the best results of prediction driving behavior. The disadvantage is that in any vehicle the placement of the smartphone can be anywhere and not only in the center console. So there should be a mechanism for virtually re-orienting the accelerometer. Also it was discovered that the average time to complete a safe lane change was 75% longer than a sudden lane change.

Chigurupati [11] et al. proposed an android-based application to aware the driver about rash driving events and to improve driver's performance with feedback. The application uses the accelerometer sensor and the GPS sensor for data recording. Also uses the camera of the smartphone for video recording. Then data analyzed to detect rash driving events. The recommended range of accelerating or braking (front/rear direction, x-axis) is -3g to +3g. The recommended range of turning, swerves or lane change (left/right direction, y-axis) is also -3g to +3g and that of bumps or road anomalies (up/down direction, z-axis) is -8g to -11g. So whenever the values of the accelerometer exceed the recommended values it would be consider as a rash driving event. The drawback of this system is that it is not automatic. There must be an administrator to analyze the videos.

Johnson [12] et al. developed an iOS application (they used an iPhone 4), which predicts and characterizes the style of the driver into normal, aggressive or very aggressive. So they built a system, which called MIROD (Mobile sensor platform for Intelligent Recognition Of Aggressive Driving). This system with the help of the application collects data from accelerometer, GPS, gyroscope, magnetometer and uses the camera for video recording. All data from multiple sensors are fused into a single classifier based on the Dynamic Time Warping (DTW) algorithm. The MIROD system can detect the following events: right or left turns (90°), U-turns (180°), aggressive right or left turns (90°), aggressive U-turns (180°), aggressive acceleration or braking, swerve right or left (aggressive lane change), device removal and excessive speed. The system can detect only aggressive events. Safe changes (for example non aggressive lane change) are not being detected because they do not exert enough force or rotation to the device. If the system detect that a driver's style becomes aggressive provides audible feedback.

Dai [13] et al. have proposed an innovative mobile phone based system, which detects drunk driving patterns. The system was implemented on Android G1 Phone and they used the accelerometer and the orientation sensor. The system can detect (throw windowing and variation thresholding) and alert the drivers about 3 categories of drunk driving behaviors. The first category is related to lane position maintenance problems like weaving, driving, swerving and turning abruptly. The second category is related to speed control problems like suddenly accelerating or decelerating, braking erratically and stopping inappropriately. The last category is related to judgment and vigilance problems like driving with tires on center or lane marker, driving without headlights at night and slow response to traffic signals. In their experiment they have tested the detection performance in abnormal curvilinear movement and problem in maintaining speed. The result

of the experiment shows 0% false negative rate and 0,49% false positive rate for abnormal curvilinear or lane changing movements. The result also shows 0% false negative and 2,39% false positive rate for speed control problems. The drawback of this system is that the set of drunk driving patterns were limited and it was difficult to distinguish them from safe and normal driving patterns.

H.Eren [14] et al. proposed an approach for estimating driving behavior and detecting if a dangerous driving pattern is safe or risky using smartphone sensors. The application of the smartphone collects data from accelerometer, gyroscope and magnetometer. Analyzing the sensor's data they obtain the speed, the position angle and the deflection from the regular trajectory of the vehicle. After the collection and the preprocessing of the sensor's data via smoothing filter they apply the endpoint detection algorithm. The endpoint detection algorithm helps to estimate the temporal range of the signal (detecting events like maneuvers). If they detect an event they use the Warping algorithm (DTW algorithm) to identify the type of the event overcoming different temporal durations of the same event across different drivers. In the end they apply Bayes Classification to identify if the event was a safe event or a risky event. In the experiment analyzed driving patterns of 15 different drivers. The results of the Bayesian Classification showed that the type of the event and if the driving style was safe or not, has been found correct for the 14 of the 15 drivers.

Chalermpol Saiprasert [15] et al. have proposed a high efficient report system using a mobile smartphone for detection and alert of dangerous over speed driving. The system collects stream of data from the smartphone's GPS sensor and produces a time series of speed and location profile for a given route. After that a speeding detection algorithm is responsible of detecting whether a vehicle over speeding by finding anomalies in speed profile. If the system detects that a vehicle is speeding can alert in real time the passengers of the vehicle. Also the system has the ability to record the data of the journey of the over speeding vehicles to be used as evidence in case a passenger wants to make a report. Findings of the experiments showed that that the system identified correctly 3 out of 4 cases of anomalies in speed profile. The data from the sensors of the smartphones are the same accurate with the data we watch in the car's speedometer. The sensitivity and the accuracy of the collected data are affected by the reception condition. Also smartphones that are from different manufacturers for the same journey and the same stream of data produce "a fluctuation in instantaneous speed measurements of approximately +4km/h due to GPS receivers with different capabilities".

Chuang-Wen You [16] et al. developed the first dual camera android application, which uses data from both cameras (front and back) of the smartphone to detect and alert drivers to dangerous driving behaviors and conditions inside and outside of the vehicle. Except the data from the cameras of the smartphone were used data of more smartphone's sensors like GPS, accelerometer and gyroscope. For the detection of dangerous driving behaviors or conditions were used computer vision and machine learning algorithms. The front camera is used to monitor and detect whether the driver of vehicle is tired or distracted. This is conducted using blink detection algorithms, which are detecting micro-sleep,

fatigue and drowsiness. The back camera is used to track road conditions, like lane change condition and the distance between cars to determine if driver's car is too close to the car in front. If the system detects any dangerous driving behaviors or conditions it alerts the driver with an audible alert and on the smartphone's screen displays an attention icon. The results of the drowsiness detection experiment showed that the detection accuracy of detecting short blinks is 88,24%, the detection accuracy of detection long blinks is 85,71% and the overall accuracy of 87,21%.

Fadi Aloul [17] et al. developed an innovative automatically notification system that uses an android based application to detect and report car accidents online. The aim of this system is to reduce fatalities by reducing the long response time required to notify emergency responders about traffic accident's data. The application collects data from the accelerometer sensor of the smartphone and then data analyzed using a Dynamic Time Warping (DTW) algorithm. After the data analysis, if there is an accident, the system detects the severity of the accident and the location of the accident using smartphone's GPS sensor. Also the system notifies police, first responders and registered emergency contact (sending an SMS) of the user's personal information (name, blood type, phone numbers of individuals etc.), the location and the severity of the accident. Findings of the experiments, testing the Dynamic time warping (DTW) algorithm showed that DTW algorithm has predict 23 out of 25 non-accident cases and all the 75 accident cases correctly. Also the experiment showed that the overall performance of the DTW algorithm in distinguishing an accident state and a non-accident state has 98% accuracy.

Nidhi Kalra [18] et al. developed an android-based application, which collects data from accelerometer sensor and then data analyzed to detect patterns of driving events and road conditions. The collected data is raw values of x,y,z axis of accelerometer sensor. After the collection, the raw values have some problems (wide range, data is noisy, etc.) so they need some preprocessing. Then gradient data analyzed to detect driving events or road anomalies. The recommended range of normal braking (in negative direction, y-axis) and sudden braking (in negative direction, y-axis) is -1g to -3g and <-3g respectively. The recommended range of sudden forward acceleration (in positive direction, y-axis) is >3g. The recommended range of left turn (in negative direction, x-axis) and right turn (in positive direction, x-axis) is <-1g and >1g respectively. Also the recommended range of pothole (change in value from positive to negative, z-axis), bump (change in value from positive to negative, z-axis) and rough road (change in value from positive to negative, z-axis) is ±1.5g.

Jin-Hyuk Hong [19] et al. have proposed an in-vehicle sensing platform that uses android smartphone's sensors and 2 other externals sensors, a Bluetooth –based on board diagnostic reader (OBD2) and an internal measurement Unit (IMU) to predict aggressive driving style. The platform with the help of the sensors collects data during driving (speed, acceleration, deceleration) and then apply machine-learning techniques with a number of driving related features for an automated assessment of driving style. The OBD2 reads data from the vehicle like engine RPM, throttle position and vehicle speed. The IMU attached to the upper back of

the steering wheel and captures the wheel movement using accelerometer, gyroscope and compass sensors. The findings of the experiment (driving data collection of 22 participants for 3 weeks) showed that the performance of the driver models, which are built using ML techniques in distinguishing an aggressive style and a non-aggressive style, has 90.5% accuracy for violation-class and 81% accuracy for questionnaire-class.

Johannes Paefgen [20] et al. have evaluated a mobile application for the assessment of driving behavior based on in-vehicle driving events and gives feedback to drivers. The implemented iOS application of the iPhone collects data from accelerometer, gyroscope and GPS sensors and then data transported to calibration component, determining the 3-dimensional orientation of the device in the vehicle. The calibration functional component contributes to the reliability and the accuracy of measurements. Then data sensors and calibrated parameters transported to trip recording component. During the trip recording sensor data is processed in a data-sampling component to detect critical driving event in real time. Users can access their data via the trip management module. Also they can receive driving feedback via the same module and share their performance to social networks. In their experiment 78 participants drove a vehicle for 45 minutes while the application was running. For the evaluation of the performance of the application, they installed inside the vehicle a dedicated off the shelf sensing system, which records reference data. Findings of the comparison of critical driving events generated by a smartphone with reference measurements from a vehicle fixed IMU showed that the measurements from the smartphone tend to overestimate critical driving events. Responsible for that is the deviation from the calibrated initial device pose. The limitation of this work is that only one model was used and threshold values were very low to achieve high-resolution model.

Magana [46] uses the light sensor in the phone to obtain information about the environment in which the car is moving, because the brightness directly affects the visibility of the driver and this influences his anticipation. Another novel method in the work of Magana is the weather information involved in estimating the driving behavior. This information is obtained from the Internet connection of the smartphone.

Araujo et al. [47] present a smartphone application, which uses the information from the embedded sensors and the vehicles state information acquired from the vehicles CAN bus (speed, fuel consumption, GPS, etc.). The gathered data is passed to a fuzzy-based module, which analyzes the data and classifies it and then a suggestion is presented to the driver how to optimize the fuel energy consumption/driving behavior.

Murphey et. al [48] propose to categorize the different driving styles according to the measure how fast the driver accelerates and decelerates. The developed algorithm extracts jerk features from the current vehicle speed within a short time-window, and classifies the current driving style into three categories: calm, normal and aggressive, by comparing the extracted jerk feature with the statistics of the driver styles on the current roadway.

Chapter 4 – Calibration of device

4.1 Vehicle Coordinate System

In order to determine driver's behavior, we need to find, how the vehicle is acting. To be able to determine how the vehicle is acting we have to collect data from smartphone's sensors. So the smartphone's collected data will represent the behavior of the vehicle. In different positions and orientations of the device inside the vehicle we get different values. In order to get always a specific range of orientation and acceleration values we have to attach our device in a fixed position (before monitoring and during our trip) and to equate the sensor's axes with the vehicle axes. This equation is done with calibration process, by reorienting smartphone's sensor's axes according to vehicle axes. So vehicle and device will have a common coordinate system, which is the coordinate system of the vehicle. We can see the 3-axes of the vehicle coordinate system in the figure below.

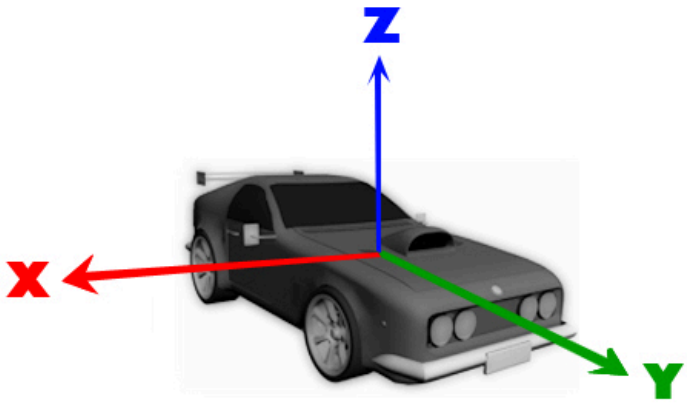


Figure 1 - Axis of Vehicle Coordinate System

4.2 Calibration

The axes of the vehicle and the axes of the smartphone or tablet are different each other as it could be seen in the figure above. In order to get accurate (acceleration or orientation) data, in each orientation in which the device is placed inside the car, we have to reorient its axes relative to the vehicle's axes.

Calibrating [21] the device relative to the vehicle means that we have to rotate the sensor's axes in order to be the same with the vehicle's axes. So we have to find the rotation angles roll, pitch and yaw. The calibration process (figure 3) of the device is carried out in 3 steps and we insure the independence of the sensor's data from the orientation in the car and the device's position.

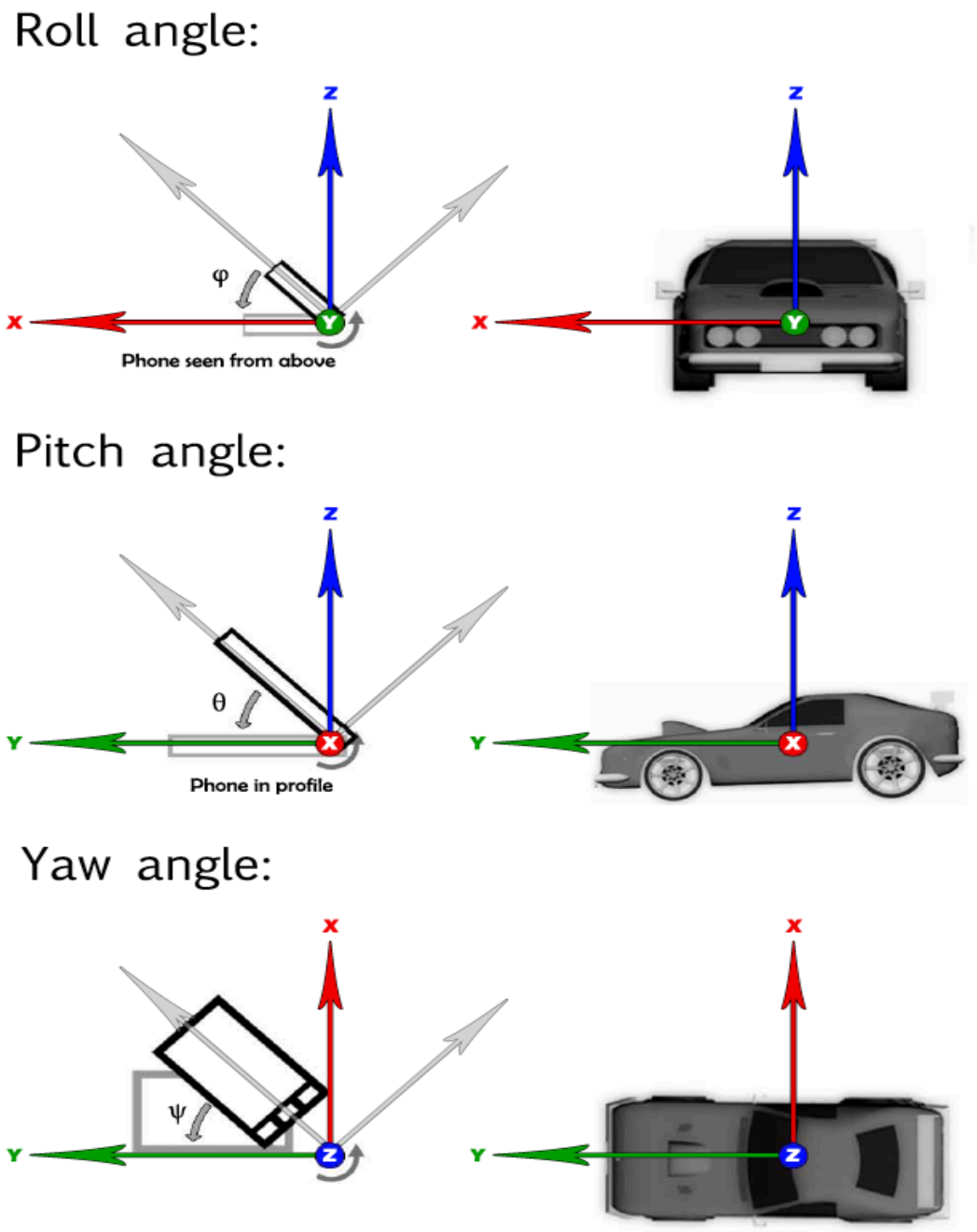


Figure 2 - Illustration of the calibration angles

During the first step the vehicle must be motionless. In the first step we have to calculate the roll and pitch rotation angle according to the vehicle's level. The roll and pitch angle are calculated with the use of the atan2 function [22] which measures the angle between two given points. To calculate the rotation angle between two axes we see the accelerometer or sensor fusion data as polar coordinates provided to the atan2 function. Below we can see the equations witch we use to calculate the roll and pitch angle.

$$roll = 2 * \arctan\left(\frac{sensorVector[2]}{sensorVector[0] + \sqrt{sensorVector[0]^2 + sensorVector[2]^2}}\right)$$

$$pitch = 2 * \arctan(\frac{sensorVector[2]}{sensorVector[1] + \sqrt{sensorVector[1]^2 + sensorVector[2]^2}})$$

Then we calculate the XY magnitude offset using the rotated sensor data (according to roll and pitch angles computed before) which is the average magnitude (30 samples) between X and Y axis, in order to check if the vehicle is in motion. If the magnitude value is relatively the same value as the last one, the vehicle is not in motion and the magnitude is added to a buffer and a counter is incremented.

$$magnitude = \sqrt{rotatedSensorVector[0]^2 + rotatedSensorVector[1]^2}$$
$$buffer = buffer + magnitude$$
$$counter = counter + 1$$
$$xyMagnitudeOffset = \frac{buffer}{counter}$$

If the vehicle is in motion we have to restart the calibration process.

In the second step the driver must start driving forward in order to calculate the yaw rotation angle. The yaw angle is the angle of the driving direction of the vehicle. First we calculate the driving direction magnitude with the help of the following equation:

$$drivingDirectionMagnitude = \sqrt{rotatedSensorVector[0]^2 - rotatedSensorVector[1]^2} - xyMagnitudeOffset$$

If the driving direction magnitude is greater than a determined threshold, the vehicle is in motion and we calculate some yaw angles using atan2 function like before.

$$yaw = 2 * \arctan(\frac{sensorVector[0]}{sensorVector[1] + \sqrt{sensorVector[1]^2 + sensorVector[0]^2}})$$

We have to compute enough angles in order to get the right direction. If we have enough angle values we get the average yaw angle and set it as the yaw angle.
In the last step we have to update the rotation angles in the module of the sensor data in order to get accurate readings. The rotated sensor data around axes calculated with the help of the following equations.

Rotated sensor data around y-axis:

$$sensorData[0] = sensorData[0] * \cos(roll) + sensorData[2] * \sin(roll)$$
$$sensorData[2] = sensorData[0] * \cos(roll) - sensorData[2] * \sin(roll)$$

Rotated sensor data around x-axis:

$$sensorData[1] = sensorData[1] * \cos(pitch) + sensorData[2] * \sin(pitch)$$
$$sensorData[2] = sensorData[1] * \cos(pitch) - sensorData[2] * \sin(pitch)$$

Rotated sensor data around z-axis:

$$sensorData[0] = sensorData[0] * \cos(yaw) + sensorData[1] * \sin(yaw)$$
$$sensorData[1] = sensorData[0] * \cos(yaw) - sensorData[1] * \sin(yaw)$$

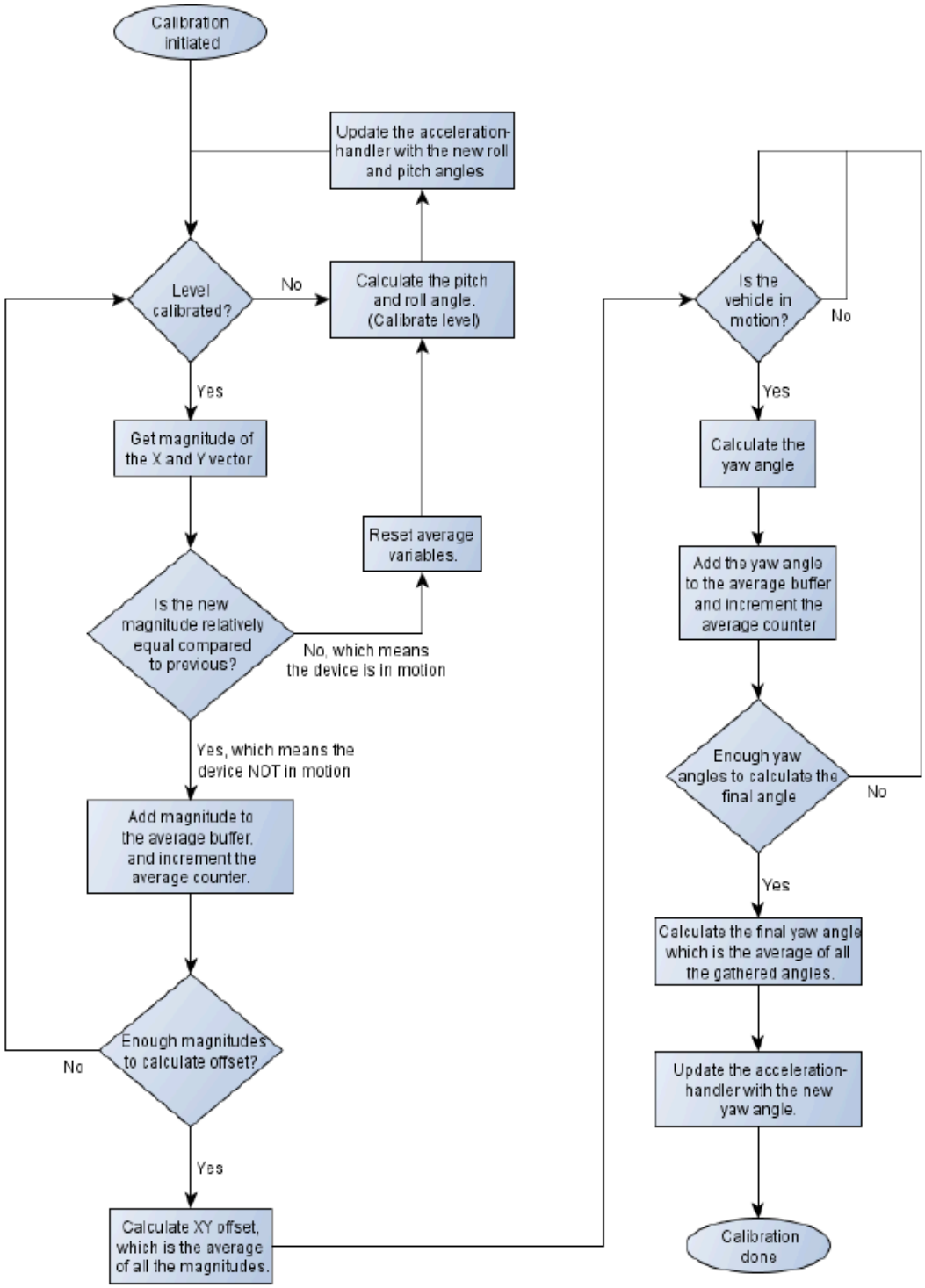


Figure 3 - Flow of the calibration process

Chapter 5 – Orientation Data via Sensor Fusion method

The best method to determine the orientation of a device (smartphone or tablet) is sensor fusion [23]. This method combines data from 3 sensors (accelerometer, magnetometer and gyroscope) in order to get the three orientation angles. We could get these three angles either using data only from accelerometer and magnetometer or using data only from gyroscope. However both ways doesn't work accurately.

Using the first way to get the orientation of the device (determine the direction of magnetic north and south), the accelerometer will provide the gravity vector (we can calculate the pitch and roll angles) and the magnetometer will work as a compass (we can calculate the azimuth angle). Both output data of sensors are inaccurate and noisy, especially the magnetometer's output data (low reaction time).

Using the gyroscope of the device we can get accurate data very fast (with short response time). The output data of the gyroscope provide the angular velocity speed for each axis. By multiplying these angular velocity measurements with the time interval between the last and the current gyroscope output data, we get the orientation angles of the device because we have a rotation increment. The main disadvantage of this way is gyro drift, which is a slow rotation of the calculated orientation. Gyro drift is caused of small errors, in each iteration of multiplying and adds up over time. This leads to a slow rotation of the calculated orientation.

So we are going back to say that the right way to get the orientation of the device is to combine all the above-mentioned sensors. The result of this combination is to avoid both noisy orientation and gyro drift. With this method counterbalance the weakness of the one sensor with the strength of the other. The low noise gyroscope output data is used only for orientation changes in short time intervals. The accelerometer and magnetometer output data is used as support information over long time intervals. With this way we filter out the gyro drift, with the accelerometer/magnetometer data, which do not drift over long time intervals. This process is equivalent to high pass filtering of the gyroscope output data and to low pass filtering of the accelerometer/magnetometer output data. This configuration called complementary filter. In the figure below we can see the overall flow of sensor fusion process. [24]

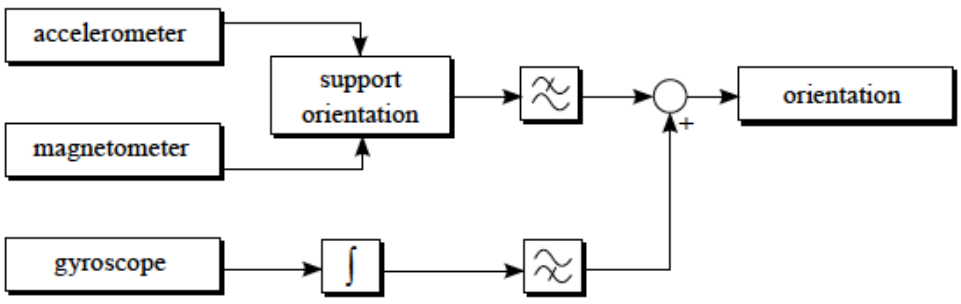


Figure 4 - Flow of the Sensor Fusion with complementary filter

All the sensors provide their output data at constant time intervals and these data can be shown as signals in a graph, with the time as the x-axis. The low pass filtering of the accelerometer and magnetometer data was done, by averaging, the orientation angle values over time within a constant time window. Every time a new accelerometer/magnetometer value is available it is weighted with a low pass factor (1-a) and added to the absolute orientation. The factor (a) is the balance factor and it's equal to 0.98. The equation for the accelerometer/magnetometer orientation is:

$$a = 0.98$$

$$AccMagOrientation = a * AccMagOrientation + (1 - a) * newAccMagOrientation$$

The high pass filtering of the gyroscope data is done by replacing, the high pass component AccMagOrientation with the corresponding gyroscope orientation data. Thus we have the final sensor fusion orientation, which is:

$$SensorFusionOrientation = a * GyroscopeOrientation + (1 - a) * AccMagOrientation$$

In the figure below we can see the intermediate signals in the filtering process, when a device is turned 90 degrees in a direction and after a short time turned back to its initial position. We can see the gyro drift in the gyroscope output data due to the small irregularities in the original angular speed. These deviations add up during the integration and cause an additional slow rotation of the gyroscope-based orientation. [25]

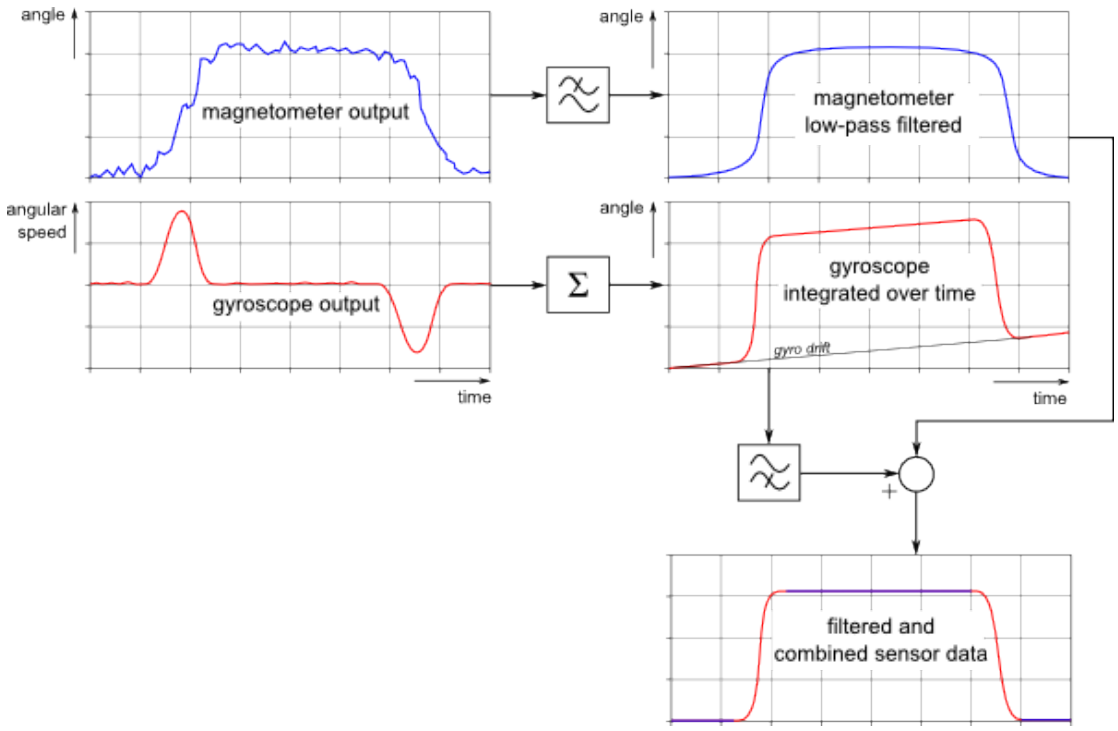


Figure 5 - Intermediate signals in the filtering process when we assuming that the device is turned 90 degrees in one direction and after a short time turned back to its initial position.

Chapter 6 – Detection Methods & Testing

For the detection of the driving behavior of the driver, we use 2 different methods. In the first detection method we use the output data of the accelerometer sensor. Collecting and evaluating the values of the accelerometer sensor we can detect how the user drives. In the second method we collect and analyze the orientation values, which we obtain with sensor fusion discussed in the previous chapter.

The thresholds acquired from the analysis of the collected data discussed in chapters 6.1 and 6.2. The collected (or monitored) values are from the accelerometer (x and y axis) and the ones from the sensor fusion (pitch and roll). We collect data using both methods during normal driving events like acceleration, turns and lane changes. Also we collect data during abnormal rash driving events and maneuvers like hard acceleration or deceleration, sharp turns and sharp lane changes.

All data from all sensors (accelerometer, magnetometer, gyroscope) of both methods is run separately through an EMA (exponential Moving Average) algorithm. [26] This algorithm works as a low pass filter and handles the noise from the sensors. We check data (output data of EMA algorithm) of both methods in time window frames of 5 seconds. In this time window all axis data of the accelerometer and of the sensor fusion are stored and analyzed. In every 5-second window frame we compute the minimum and the maximum value of all axis data.

The collection of the driving events was conducted in areas where there is not traffic at all. Also for the collection and testing of data were used 3 different devices. A Samsung Galaxy tablet (2014 edition), a Samsung Galaxy s3 smartphone (2013 edition) and a LG G3 (2014 edition). The collected data from the all devices match each other. So the sensor's output data more or less are the same for different devices. The device was placed in a fixed position in the central console of the vehicle. After the calibration and while monitoring you cannot move the device from its fixed position. If the device moved while monitoring the sensor's output data will be wrong and the evaluation of the data will not be accurate.

In the next chapters and for both detection methods, we can see which sensor data we use to detect for every driving event. Also we can see graphs for every driving event where the maneuver is clearly distinguished.

6.1 Detection Method using Accelerometer Data

In this method we have used the three-axis accelerometer sensor to record and analyze various driving events and driver's behavior. We have utilized x- axis data to detect the left/right direction of the vehicle and therefore driving events like safe or sharp turns and safe or sharp lane changes. For the detection of front/rear direction of vehicle and therefore to measure how the driver accelerate and apply the brakes we have utilized y-axis data.

After collecting and analyzing our data we determined the threshold values of various events and maneuvers. Acceleration or deceleration of the vehicle is determined by the change in acceleration in y-axis. Safe acceleration is considered when the y-axis value is between 1.3 m/s^2 and 2.5 m/s^2. When the value is more than 2.5 m/s^2 it is considered as hard (sudden) acceleration. Similar to acceleration, it is considered safe deceleration (normal braking) when we have a value between -1.3 m/s^2 and -2.5 m/s^2. When the y-axis value is lower than -2.5 m/s^2 it is considered as hard deceleration (sudden braking). In the following table and graphs we can see the thresholds and patterns for acceleration/deceleration driving event.

Table 1 - Thresholds and Data used for the detection of a Safe/Hard - Acceleration/Deceleration using accelerometer's data.

Driving event	Data used for detection	Threshold
Safe Acceleration	Y-axis data	1.3 m/s^2 to 2.5 m/s^2
Hard Acceleration	Y-axis data	> 2.5 m/s^2
Safe Deceleration	Y-axis data	-1.3 m/s^2 to -2.5 m/s^2
Hard Deceleration	Y-axis data	< -2.5 m/s^2

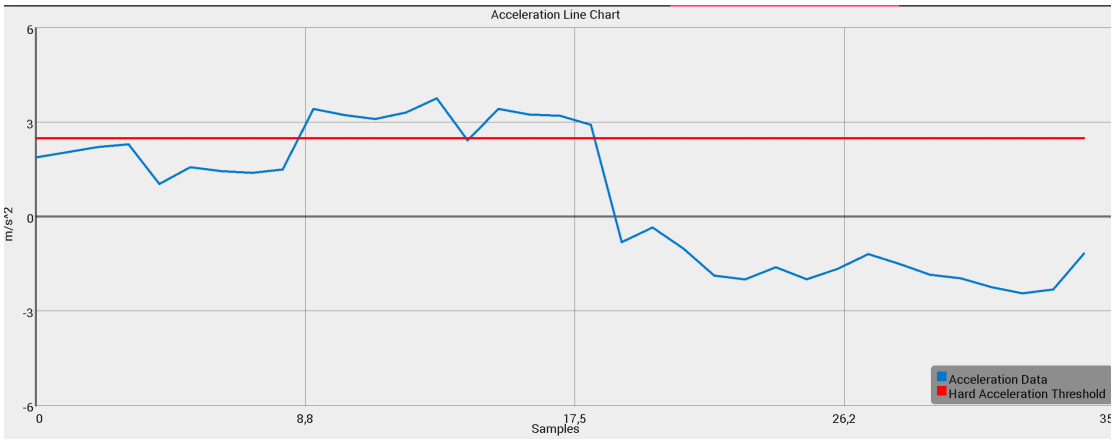


Figure 6 - Safe Acceleration pattern and then a hard acceleration pattern.

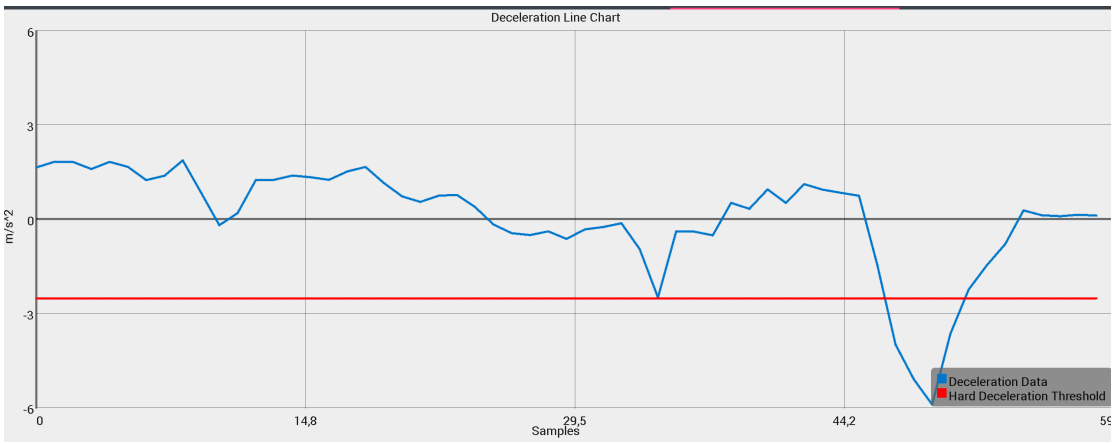


Figure 7 - Safe Deceleration pattern and then a hard deceleration pattern

Left or right turn is determined by the change in acceleration in x-axis. Safe left turn is considered when we have x-axis value between -1.8 m/s^2 and -3 m/s^2.

Whenever the accelerometer x-axis value exceeds the upper safe limit (-3 m/s^2), it would be considered as sharp left turn. Similar to left turn, we have considered safe right turn when the acceleration in x-axis has value from 1.8 m/s^2 to 3 m/s^2 and sharp right turn when the value exceeds the upper safe limit (3 m/s^2). In the following table and graphs we can see the thresholds and patterns for safe/sharp left/right turn driving event.

Table 2 - Thresholds and data used for the detection of a Safe/Sharp - Left/Right Turn using accelerometer's data.

Driving event	Data used for detection	Threshold
Safe Left Turn	X-axis data	-1.8 m/s^2 to -3.0 m/s^2
Sharp Left Turn	X-axis data	< -3.0 m/s^2
Safe Right Turn	X-axis data	1.8 m/s^2 to 3.0 m/s^2
Sharp Right Turn	X-axis data	> 3.0 m/s^2

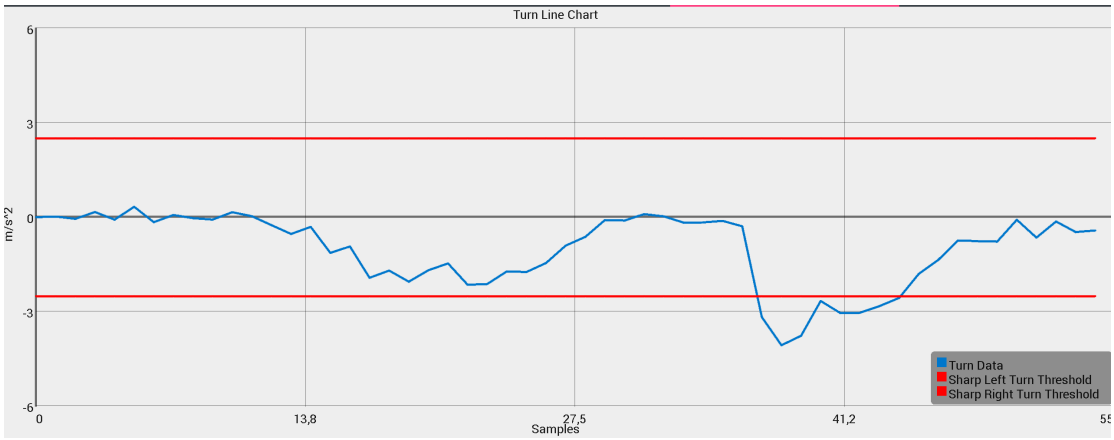


Figure 8 - Safe Left Turn pattern and then a Sharp Left Turn pattern.

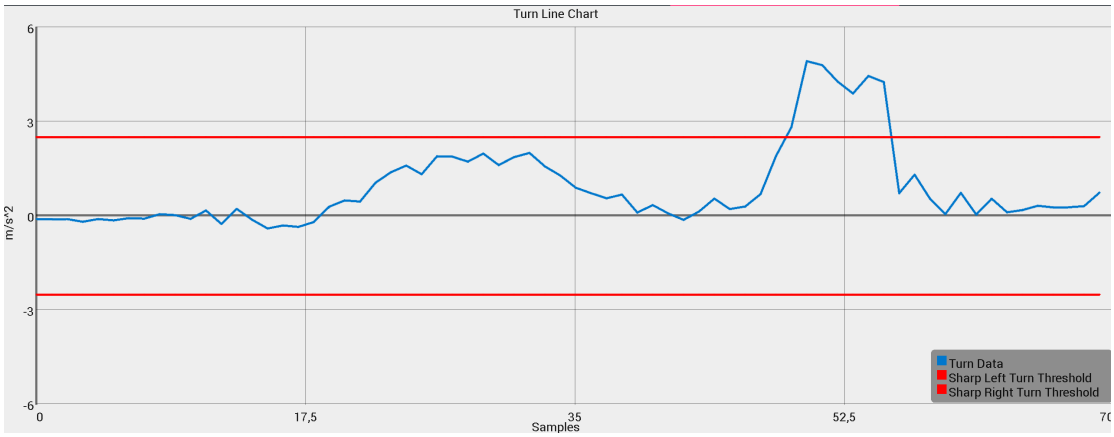


Figure 9 - Safe Right Turn pattern and then a Sharp Right Turn pattern.

A lane change also is determined by the change in acceleration in x-axis and by the number and the type of turns in a specific time. Safe right lane change is considered when we have a safe right turn and in the next 2-second time window frame we have a safe left turn. Safe left lane change is considered when in a 2

second time window occurred two safe turns. The first must be safe left turn and the second, safe right turn. On the other hand, in the case of sharp lane changes we have 2 cases. In the first case one of the turns must be sharp and in the second case both of them must be sharp. For example we have a sharp right lane change when occurred 2 turns (the first right and the second left) in a 2 second time window and the first turn is a safe turn and the second a sharp turn or the first turn is sharp and the second is safe turn. Also a sharp right lane change considered when both of the turns are sharp, the first one is sharp right turn and second one is sharp left turn.

In the graphs we can see the thresholds and patterns for safe/sharp lane change driving event.

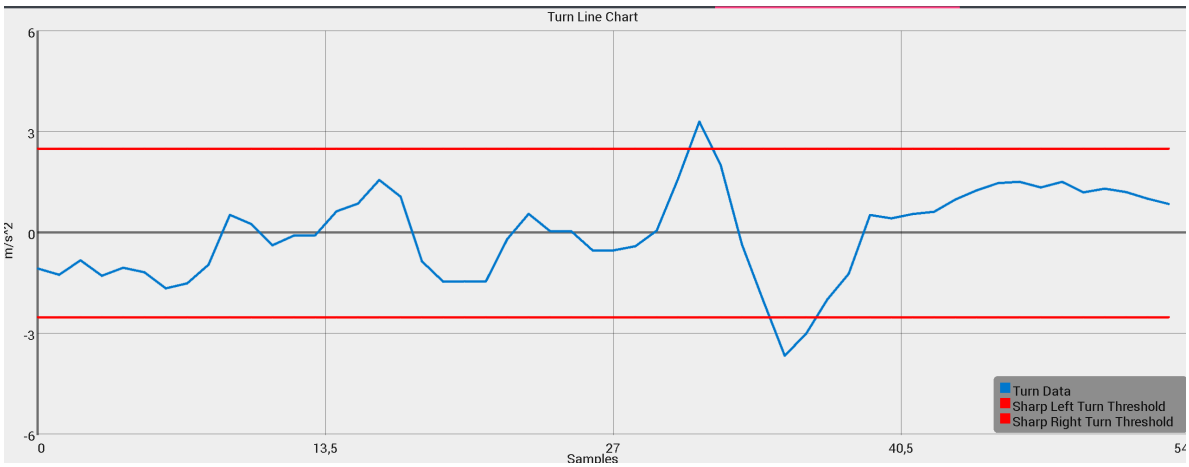


Figure 10 - Safe Right Lane Change pattern and then a Sharp Right Lane Change Pattern.

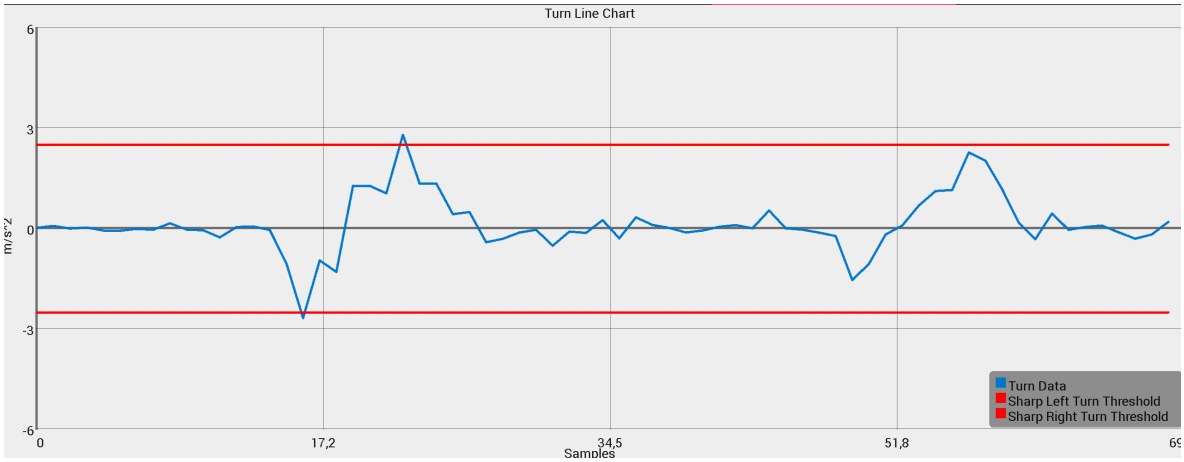


Figure 11 - Safe Left Lane Change pattern and then a Sharp Left Lane Change pattern.

6.2 Detection Method using Sensor Fusion Orientation Data

In this method we have used the three-axis orientation output data of sensor fusion in order to collect and analyze various driving events. Orientation is defined as a combination of three angular quantities: azimuth, pitch and roll.

These three quantities are defined based on 3-axis. The positive X-axis extends out of the right side of the car, positive Y-axis extends out of the front side of the car and the positive Z-axis extends out of the topside of the car. Azimuth is angle between the positive Y-axis and magnetic north and its range is between 0 and 360 degrees. Positive roll is defined when the car starts by laying flat on the road and the positive Z-axis begins to tilt towards the positive X-axis. Positive pitch is defined when the car starts by laying flat on the road and the positive Z-axis begins to tilt towards the positive Y-axis.

We have utilized roll data to detect the left/right direction of the vehicle and therefore driving events like safe or sharp turns and safe or sharp lane changes. For the detection of front/rear direction of vehicle and therefore to measure how the driver accelerate and apply the brakes we have utilized pitch data.

After collecting and analyzing our data we determined the threshold values of various events and maneuvers. Acceleration or deceleration of the vehicle is determined by the change in orientation in pitch angle. Safe acceleration is considered when the pitch value is between -0.08 rad/s and -0.12 rad/s. When the value is less than -0.12 rad/s it is considered as hard (sudden) acceleration. Similar to acceleration, it is considered safe deceleration (normal braking) when we have a value between 0.08 rad/s and 0.12 rad/s. When the pitch value is higher than 0.12 rad/s it is considered as hard deceleration (sudden braking). In the following table and graphs we can see the thresholds and patterns for acceleration/deceleration driving event of sensor fusion method.

Table 3 - Thresholds and Data used for the detection of a Safe/Hard - Acceleration/Deceleration using orientation data of Sensor fusion method.

Driving event	Data used for detection	Threshold
Safe Acceleration	Pitch angle	-0.08 to -0.12 rad/s
Hard Acceleration	Pitch angle	< -0.12 rad/s
Safe Deceleration	Pitch angle	0.08 to 0.12 rad/s
Hard Deceleration	Pitch angle	> 0.12 rad/s

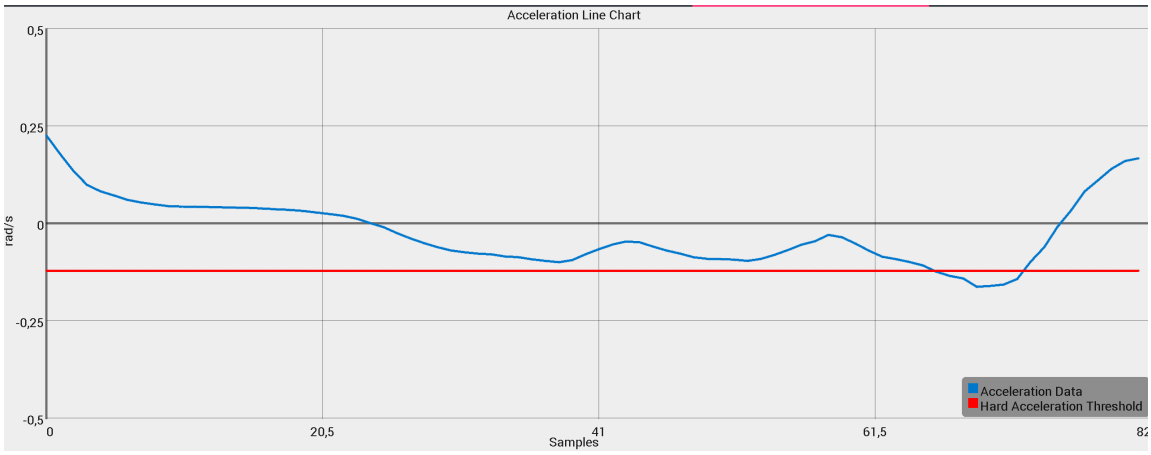


Figure 12 - Two safe acceleration patterns and then a hard acceleration pattern.

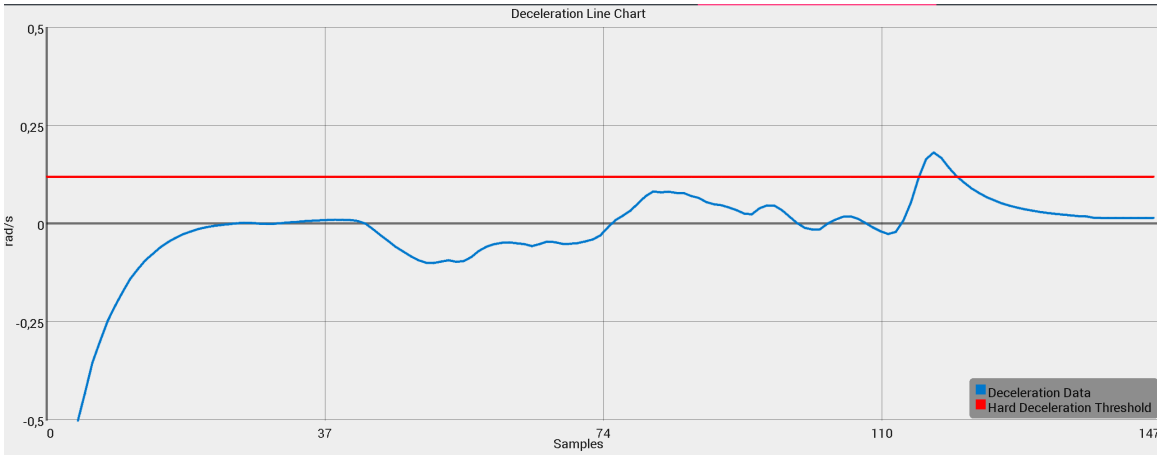


Figure 13 - A safe deceleration pattern and the a hard deceleration pattern.

Left or right turn is determined by the change in orientation in roll angle. Safe left turn is considered when the roll value is between 0.10 rad/s and 0.30 rad/s. Whenever the roll value exceeds the upper safe limit (0.30 rad/s), it would be considered as sharp left turn. Similar to left turn, we have considered safe right turn when the roll angle has value from -0.10 rad/s to -0.30 rad/s and sharp right turn when the value exceeds the upper safe limit (-0.30 rad/s). In the following table and graphs we can see the thresholds and patterns for safe/sharp left/right turn driving event of sensor fusion method.

Table 4 - Thresholds and Data used for the detection of a Safe/Sharp - Left/Right Turn using orientation data of the sensor fusion method.

Driving event	Data used for detection	Threshold
Safe Left Turn	Roll angle	0.10 to 0.30 rad/s
Sharp Left Turn	Roll angle	> 0.30 rad/s
Safe Right Turn	Roll angle	-0.10 to -0.30 rad/s
Sharp Right Turn	Roll angle	< -0.30 rad/s

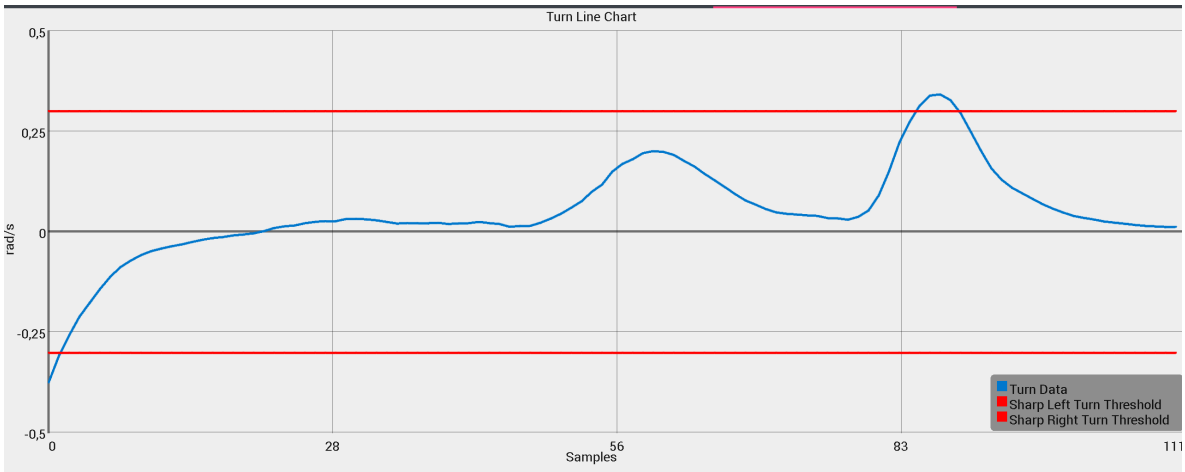


Figure 14 - A Safe Left Turn pattern and then a Sharp Left Turn pattern

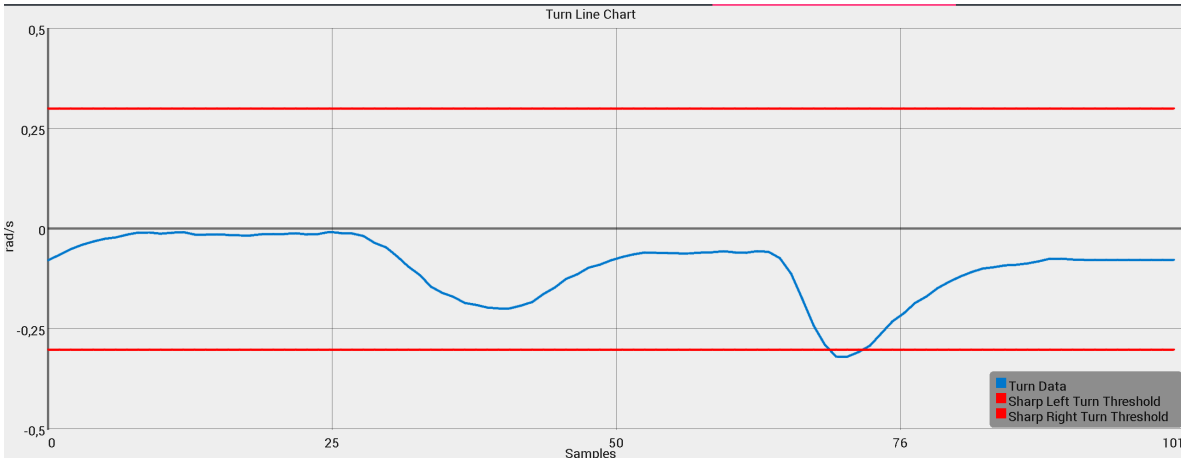


Figure 15 - A Safe Right Turn pattern and then a Sharp Right Turn pattern.

In case we want to detect safe and sharp lane changes using orientation data, the philosophy and the methodology is the same as in the previous method, detection using accelerometer data. A lane change also is determined by the number and the type of turns in a specific time. Safe right lane change is considered when we have a safe right turn and in the next 2-second time window frame we have a safe left turn. Safe left lane change is considered when in a 2 second time window occurred two safe turns. The first must be safe left turn and the second, safe right turn. On the other hand, in the case of sharp lane changes we have 2 cases. In the first case one of the turns must be sharp and in the second case both of them must be sharp. For example we have a sharp right lane change when occurred 2 turns (the first right and the second left) in a 2 second time window and the first turn is a safe turn and the second a sharp turn or the first turn is sharp and the second is safe turn. Also a sharp right lane change considered when both of the turns are sharp, the first one is sharp right turn and second one is sharp left turn.

In the graphs we can see the thresholds and patterns for safe/sharp lane change driving event.

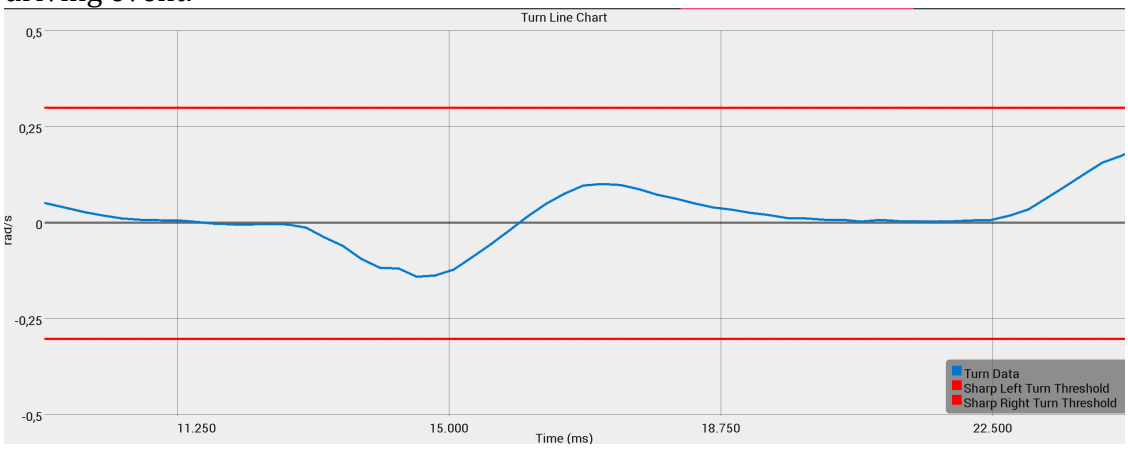


Figure 16 - A Safe Right Lane Change Pattern.

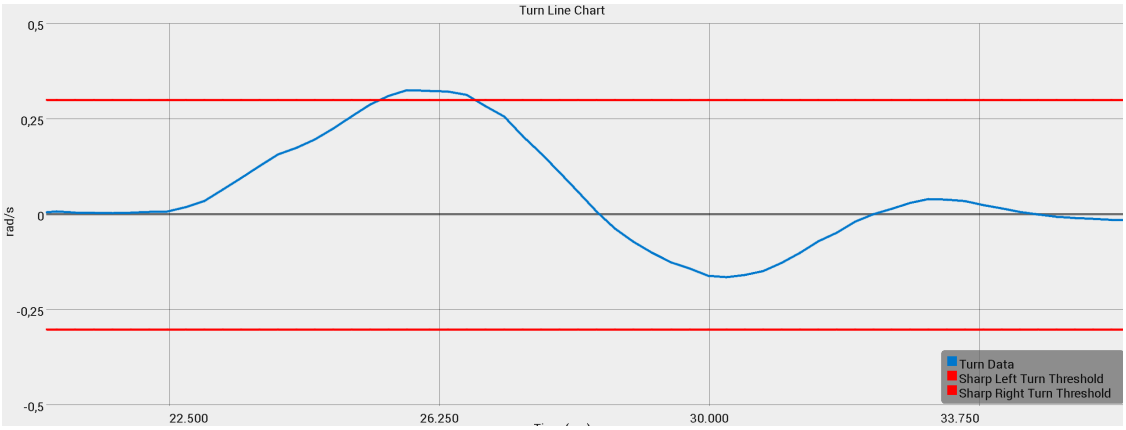


Figure 17 - A Sharp Left Lane Change pattern.

6.3 Driver Behavior Detection Algorithm

Our algorithm characterizes the behavior of the driver as *Excellent*, *Very Good*, *Good*, *Bad* or *Very Bad* and computes the average speed of the vehicle at the end of every trip.

Our algorithm for the detection of driver behavior uses two types of data. The acceleration output data of the accelerometer sensor and the orientation output data of the sensor fusion method. We can detect the behavior of the driver using either acceleration data, either orientation data. The driver can choose which detection method will be used for the detection of his behavior. It can be choosed only one method (type of data). The driver behavior detection algorithm doesn't work, with the use of both data types at the same time.

After we choose detection method, the algorithm analyzes the behavior of the driver based on thresholds of detection data. These thresholds are acquired by testing the data of the detection methods under various driving events and maneuvers. Detection methods and testing is discussed in the chapter 6. With these thresholds we can distinguish and detect 12 driving events. Six of them are safe driving events and the other six are dangerous driving events. The safe events are: Safe Acceleration, Safe Deceleration, Safe Left Turn, Safe Right Turn, Safe Left Lane Change and Safe Right Lane Change. The dangerous events are: Hard Acceleration, Hard Deceleration, Sharp Left Turn, Sharp Right Turn, Sharp Left Lange Change and Sharp Right Lane Change.

Our algorithm for the detection of the driver's behavior, works in time window frames of 5 seconds. In these 5 seconds all data (3-axis values) of the detection method (accelerometer or sensor fusion data) are stored in a list. In every time window frame, new values are inserted in the list and the same time the oldest values are removed from the same list. Then we check every data (3-axis value) of the list. If the current value, used for the detection of maneuvers is below the threshold of safe driving events, means that none of various maneuvers the system can detect is happening. When the current value is bigger than the threshold for safe driving events and below the threshold for dangerous driving events means that, one of the safe-driving events is happening. At last if the

monitored current value exceeds the threshold for a dangerous maneuver means that we have a dangerous situation like sharp right turn.

In case of Lane Changes the algorithm checks if in time window frame of 2 seconds have occurred two turns. One left and one right or the opposite. If one of them turns or both of are sharp turns, which means that the value, used for the turn detection of the car is higher than the threshold for a dangerous turn, a sharp lane changing is happening. On the other hand if both of them are safe turns, a safe lane changing is happening. If the first turn is left turn we have a left lane change and if it is right we have a right lane change.

In the table below we can see the summary thresholds of both methods, which used for the detection of various *safe* and *dangerous* maneuvers.

Table 5 - Thresholds and Data Used for the detection of various driving events using accelerometer's data or sensor fusion orientation data.

Driving Event	Data Used (Accelerometer)	Threshold	Data Used (Sensor Fusion)	Threshold
Safe Acceleration	Y-axis data	1.3 m/s^2 to 2.5 m/s^2	Pitch angle	-0.08 to -0.12 rad/s
Safe Deceleration	Y-axis data	-1.3 m/s^2 to -2.5 m/s^2	Pitch angle	0.08 to 0.12 rad/s
Safe Left Turn	X-axis data	-1.8 m/s^2 to -3.0 m/s^2	Roll angle	0.10 to 0.30 rad/s
Safe Right Turn	X-axis data	1.8 m/s^2 to 3.0 m/s^2	Roll angle	-0.10 to -0.30 rad/s
Hard Acceleration	Y-axis data	> 2.5 m/s^2	Pitch angle	< -0.12 rad/s
Hard Deceleration	Y-axis data	< -2.5 m/s^2	Pitch angle	> 0.12 rad/s

Sharp Left Turn	X-axis data	< -3.0 m/s^2	Roll angle	> 0.30 rad/s
Sharp Right Turn	X-axis data	> 3.0 m/s^2	Roll angle	< -0.30 rad/s

Every driving event or maneuver, that our system can detect, has a counter. When the algorithm detects that the driver makes one of the already talked maneuvers or driving events, the counter for this type of event is incremented. If the algorithm detects a safe driving event, a counter for this driving event is incremented. When detects a dangerous driving event, then a counter for this event is incremented as well.

When the driver finished his trip. Our algorithm computes the percentage of the penalty for every dangerous driving event. The computed penalties are: hard acceleration penalty, hard deceleration penalty, sharp left turn penalty, sharp right turn penalty, sharp left lane change penalty and sharp right lane change penalty.

The equations for the Hard Acceleration and Hard Deceleration penalties are:

$$HardAccelerationPenalty = \frac{HardAccelerationCounter}{SafeAccelerationCounter + HardAccelerationCounter}$$

$$HardDecelerationPenalty = \frac{HardDecelerationCounter}{SafeDecelerationCounter + HardDecelerationCounter}$$

The equations for the Sharp Left Turn and Sharp Right Turn are:

$$SharpLeftTurnPenalty = \frac{SharpLeftTurnCounter}{SafeLeftTurnCounter + SharpLeftTurnCounter}$$

$$SharpRightTurnPenalty = \frac{SharpRightTurnCounter}{SafeRightTurnCounter + SharpRightTurnCounter}$$

The equations for the Sharp Left Lane Change and Sharp Right Lane Change are:

$$SharpLeftLaneChangePenalty = \frac{SharpLeftLaneChangeCounter}{SafeLeftLaneChangeCounter + SharpLeftLaneChangeCounter}$$

$$SharpRightLaneChangePenalty = \frac{SharpRightLaneChangeCounter}{SafeRightLaneChangeCounter + SharpRightLaneChangeCounter}$$

Using the above penalties we can compute the Total Sharp Turn Penalty and the Total Sharp Lane Change Penalty with the following equations:

$$SharpTurnPenalty = SharpLeftTurnPenalty + SharpRightTurnPenaly$$

$$SharpLaneChangePenalty = SharpLeftLaneChangePenalty + SharpRightLaneChangePenalty$$

The equation for the total penalty is:

$$TotalPenalty = HardAccelerationPenalty + HardDecelerationPenalty + HardTurnPenalty + HardLaneChangePenalty$$

Except the penalties at the end of the trip, the algorithm computes the driving score of the driver. The total score depends of the penalties of the dangerous driving events (Total Penalty). The maximum score a user can achieve is ten (10) and the minimum is zero (0).

The equation for the computation of the total score, achieved from the user at the end of the trip is:

$$TotalScore = 10 - TotalPenalty$$

According to the total score (at the end of the trip) of the user, the algorithm characterizes the user for his driving behavior for the current trip. There are various behaviors depending the total score of the user. The behavior of the driver at the end of the trip can be *Excellent*, *Very Good*, *Good*, *Bad* or *Very Bad*. The various driver behaviors determined in the table below.

Table 6 - Driver Behavior categories based on the total score of the driver.

Driving Behavior	Total Score
Excellent	Score>9.75
Very Good	9<Score<=9.75
Good	7.5<Score<=9
Bad	5<Score<=7.5
Very Bad	Score<=5

For the computation of the average speed (avgSpeedKM) of the vehicle at the end of every trip, we have to compute first the total time (totalHours) we were driving and the total distance (distanceKM) we traveled. When we start a new trip the startTrip() function is called. During the trip, the total distance travelled is computed inside the updateLocation() function. This function is called every time we have a new Location from GPS. When the trip is finished the stopTrip() function called and the average speed is computed by dividing the total distance with the total time.

Below we can see the functions, which are called for the computation of the average speed of the vehicle for every trip.

```
updateLocation(Location location)
{
      // if there is a previous location
      if (previousLocation != null) {
        // add to the total distanceTraveled
        distanceTraveled += location.distanceTo(previousLocation);
      } // end if

      previousLocation = location;
}


startTrip()
{
    driving = true;
    startTime = System.currentTimeMillis(); // get current time
    previousLocation = null; // starting a new trip
}


stopTrip()
{
    driving = false; // just stopped tracking locations
    MILLISECONDS_PER_HOUR = 1000 * 60 * 60;

    // compute the total time we were driving
    long milliseconds = System.currentTimeMillis() - startTime;
    double totalHours = milliseconds / MILLISECONDS_PER_HOUR;

    double distanceKM = distanceTraveled / 1000.0;
    double avgSpeedKM = distanceKM / totalHours;
}
```

Chapter 7 – Information System of Usage-Based Auto Insurance

7.1 Native Android Application of Auto UBI System

We developed a native android-based application that can be used by a usage-based insurance company. The application via smartphone's sensors can detect and evaluate the driving behavior of the user for all his trips. All trip data that contains statistics, routes and graphs is saved to in smartphone's local memory. Also these data is sent to a company's server, where can be accessed by the employees of the company. The company evaluates the data of all the trips of the user, whose payment to the company is depending to his average behavior and to total distance of his trips. In the subsections below we can see how the application works, what data are recorded and how they presented to the user.

7.1.1 - Driver's Login and Registration System

After the user installs and enters the application in his device, the first thing that he sees is a login system. If the user has already registered in the system, by entering his e-mail and password in the respective fields, can be entered into the system. If is not registered in the system, he has to sign up by pressing the "Be better driver! Sign up now" button. In the registration form has to enter the following data: Full name, e-mail, password, address, phone number and the vehicle license plate. When the user completes his registration process he can login successfully in the system. In the figures bellow, we can see the login and the registration screens.

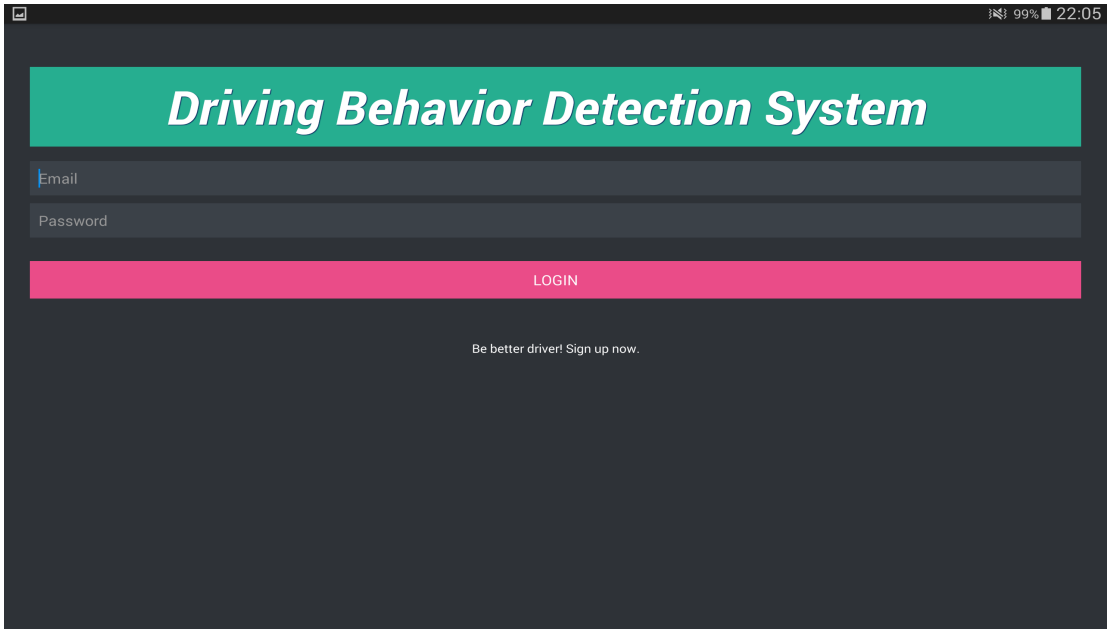


Figure 18 - The Login Screen of the application.

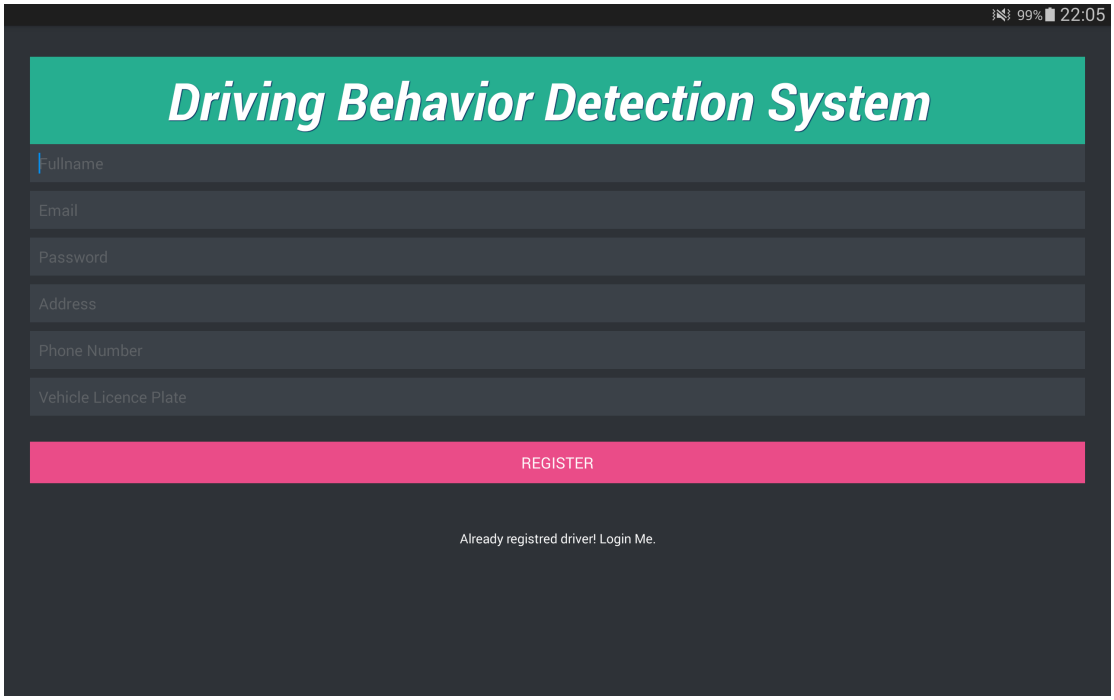


Figure 19 - The Registration screen of our application.

7.1.2 - Main Menu

When the user successfully connected, the main menu is presented. In the main menu there are 4 options. The first option is the "New Trip". We choose this option when we want to start a new trip. The second option is the "My trips" and we choose it when we want o review our trips (routes and statistics). In the third option there are our settings and the last option is "Help" where we can find tutorials and information about how we use the application. We can see the main menu in the figure below.

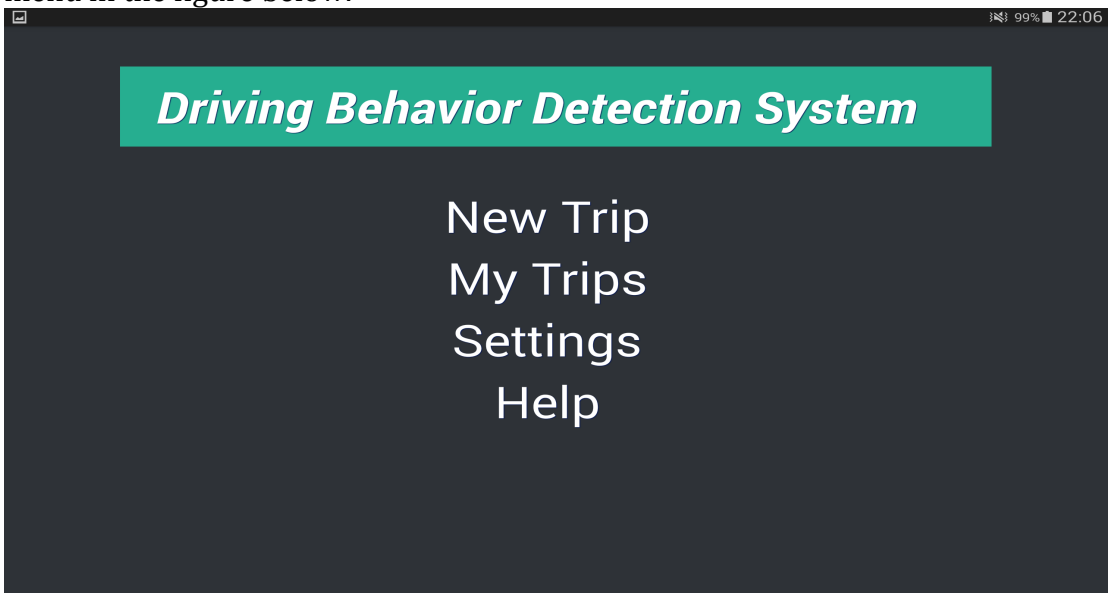


Figure 20 - The Main Menu of our application.

7.1.3 - New Trip

As already motioned above, we choose the "New Trip" option when we want to start a new trip. Before we start driving the application will tell us to follow some instructions for the calibration of the device. We set the calibration procedure for the device in order to get accurate readings from the sensor's data in any fixed orientation of the device inside the vehicle. In the figure below we can see the instructions for the calibration of the device.

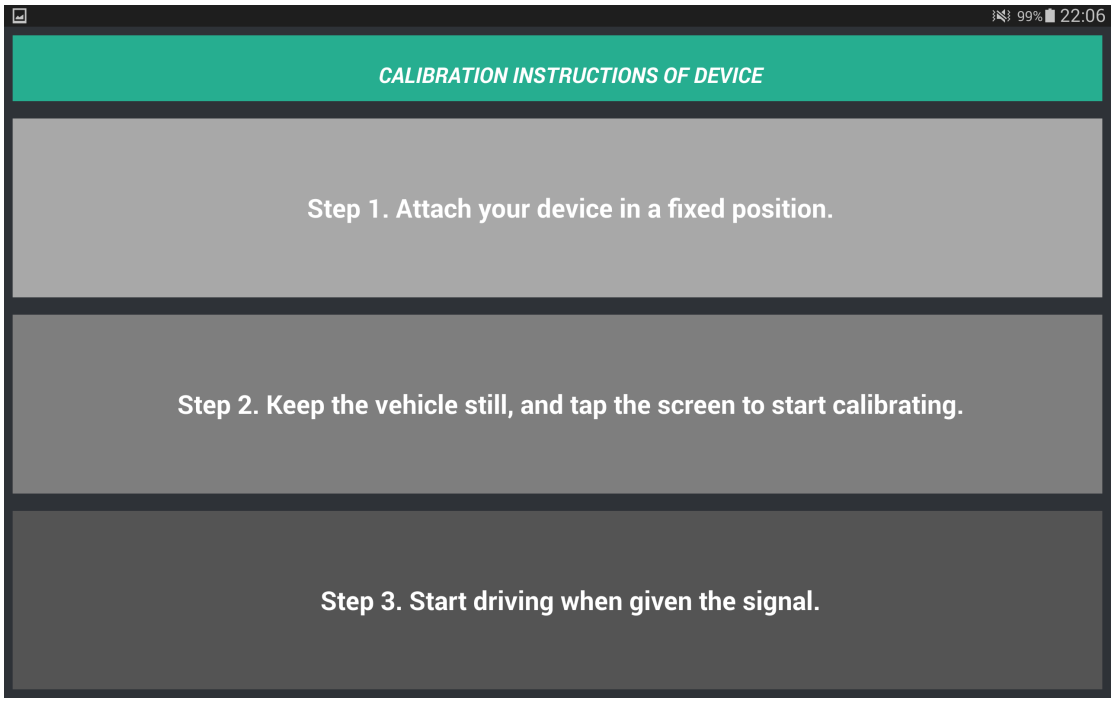


Figure 21 - The calibration instructions screen.

During the calibration process the device must be attached in a fixed position and the vehicle must be still. The whole process takes about 5 seconds to complete. When the signal is given we start driving to our destination. We can see the given signals in the figure bellow.

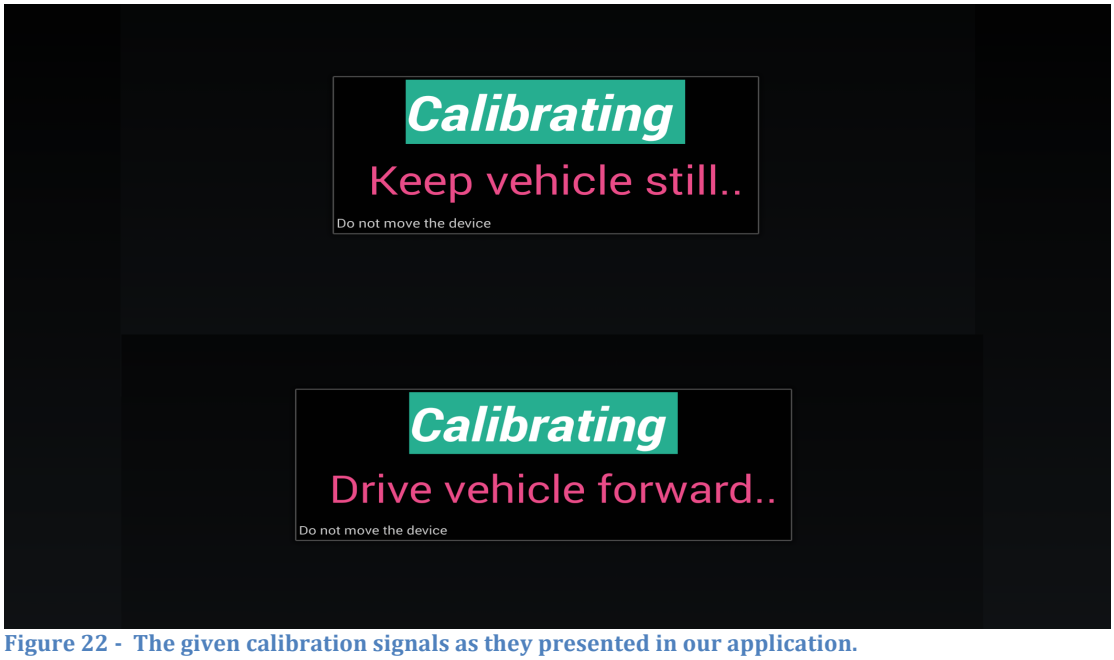


Figure 22 - The given calibration signals as they presented in our application.

In the beginning of the process the signal to keep vehicle still is displayed and in a few seconds “Drive vehicle forward ” signal is displayed. When we start driving forward the calibration completed and starts monitoring and detecting our driving behavior until the end of our trip.

The user can select between two options of displaying the monitoring of the driving behavior. The first option is the basic option where in the main screen there is displayed a “Monitoring Driving Behavior” message.

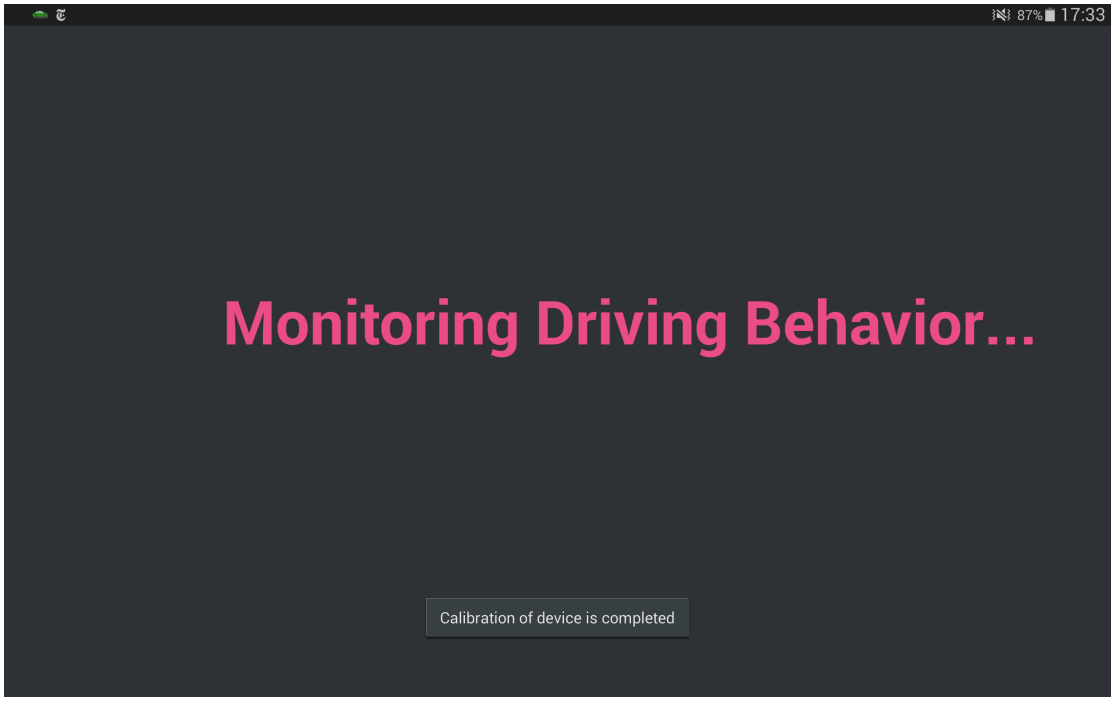


Figure 23 - The basic monitoring option. The system is not detecting anything safe or dangerous driving events.

When the system detects a safe driving event (maneuver) the main screen color changed to green color and the message also changed to the name of the current safe driving event.

Figure 24 - The system detects a safe deceleration.

When the system detects a dangerous driving event the color of the main screen changed to yellow color and the message also changed to “Attention!” followed by the name of the dangerous event. For all the dangerous driving events except the attention and the name of the event it is displayed a hint. The hint helps the driver to improve his driving skills. The hints help drivers to improve their driving behavior and to achieve better scores in their trips.

Also by providing constructive feedback to drivers is very important and helps them to correct bad driving behaviors. The hints that are displayed are: try to maintain uniform acceleration (or deceleration), try to take left (or right) turn slower and try to change the left (or right) lane slower. The attention message and hint is followed by a notification sound. In the figure below we can see the screen of out device when a sharp left turn is detected.

Figure 25 - The system detects a dangerous sharp left turn.

The second option of displaying the monitoring of driving detection of the user is the Map option. With this option in our screen is displayed a map with our spot (the car’s spot) into the map. When we start driving, our route outlined in the map. Also it is displayed with pointers, the starting point of our trip and any bad driving event (maneuver) is going to happen.

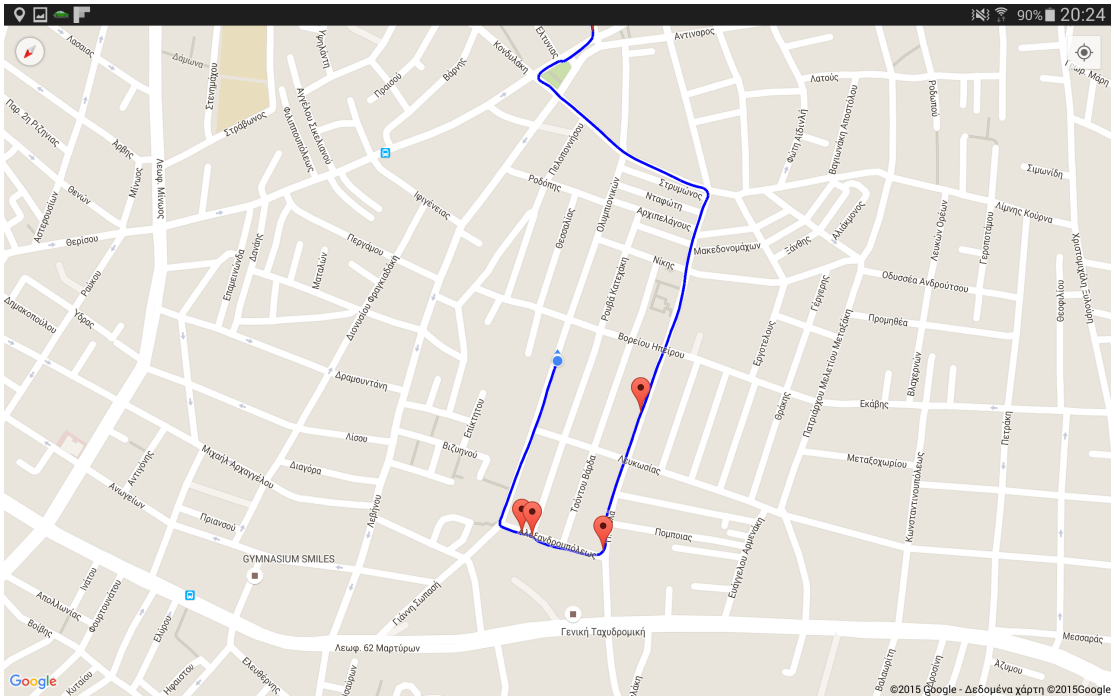


Figure 26 - The map monitoring option of our application.

When the system detects a safe driving a pop up window is opening and displays the current safe driving event. In the same way when a dangerous driving event is detected a pop-up window with the name and the hint of the event is displayed for some seconds. Also a notification sound listened and a pointer in the map with the name of the dangerous driving event is created.

In the figure below we can see the screen of out device with the map option of displaying the monitoring of the driving behavior, when a sharp left turn is detected.

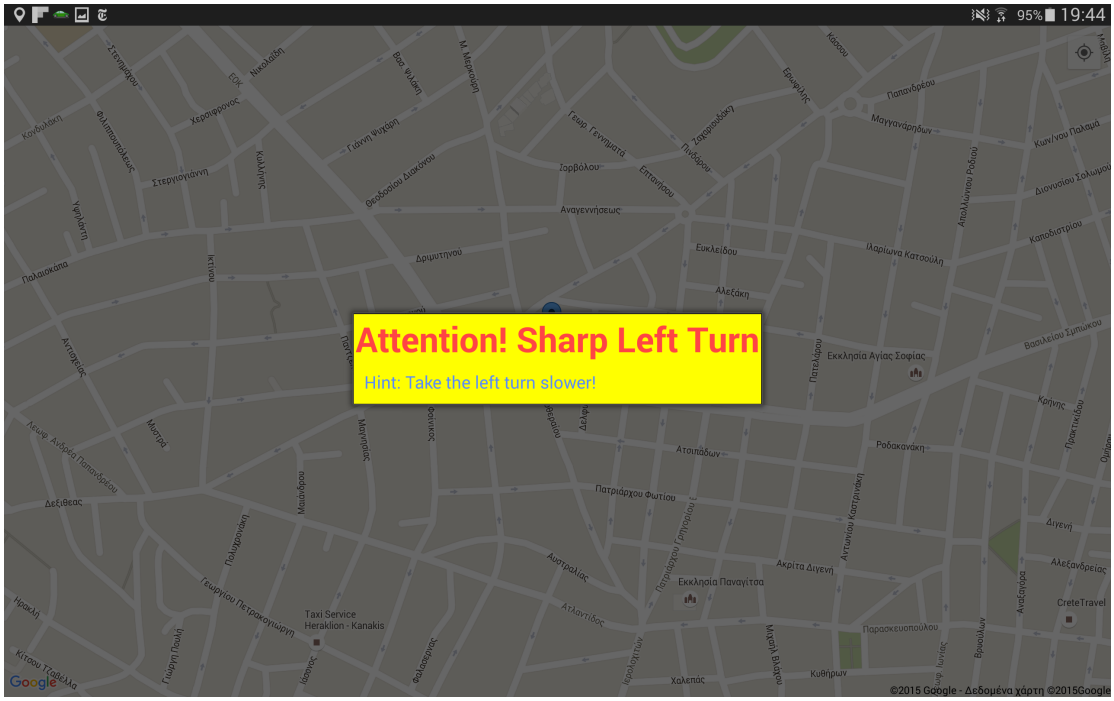


Figure 27 - The system detects a sharp left turn during the map monitoring option.

We can alternate between the 2 options of displaying the monitoring of our driving behavior (Basic Monitor to Map Monitor and the opposite) any time we want during the trip by pressing the option button of our device. When we press the option button, an option menu is displayed in the bottom of our screen and we can choose out choice. When the user finished his trip, he presses the back button on his device and the system ends the trip and loads all the statistics, the routes and the graphs. In the figure below we can see the option menu for changing monitor.

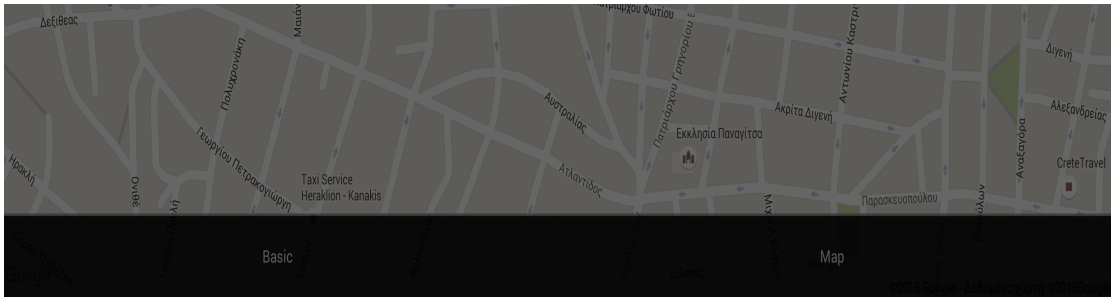


Figure 28 - Option menu, where we can select the monitoring option we prefer.

7.1.4 - Trip's Info

As we mentioned before, by pressing the back button of our device we finished our trip. After that the system is loading and presented all information about our trip. All information of the current trips is presented in 3 different tabs.

In the first tab is presented the info tab. In info tab presented some general statistics about the trip like the total distance, the total duration and the average speed. The total duration is the actual duration of the trip. Except the above statistics is presented the percentage of the penalties of dangerous driving events. The presented percentages of penalties are: Hard Acceleration Penalty, Hard Deceleration Penalty, Sharp Left Turn Penalty, Sharp Right Turn Penalty, Sharp Right Left Lane Change Penalty and Sharp Left Lane Change Penalty. Also is present the rating (Score - up to 10) and the behavior (Very Bad to Excellent) of the current trip. In the following figure we can see the Info Tab.

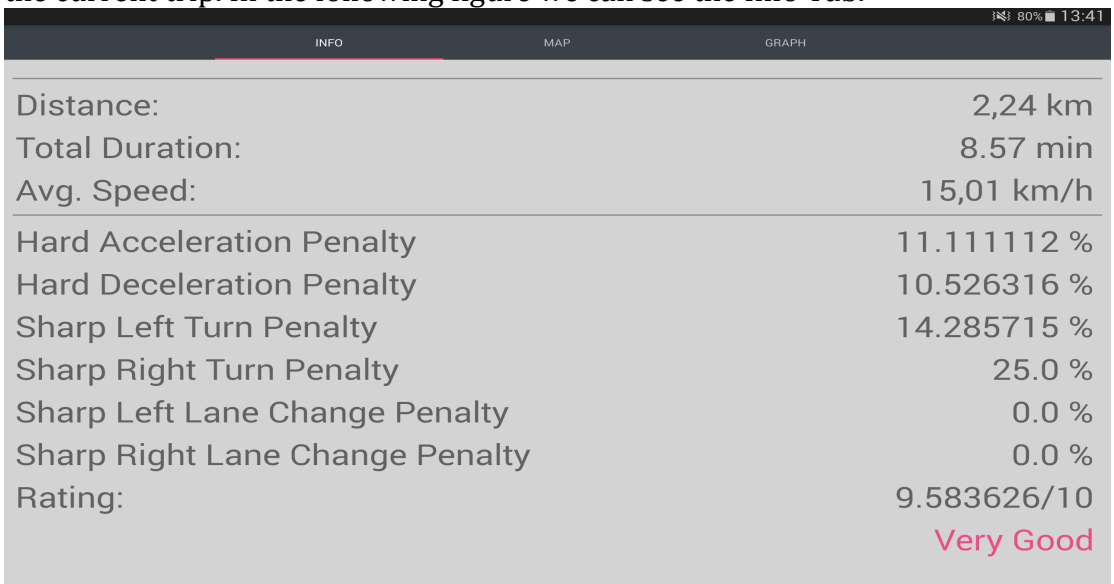


Figure 29 - Info Tab screen of our application.

In the second tab is presented the map tab. In map tab is presented the route of our trip based on Google maps. In the map we can see the starting point of our trip, which is represented by a blue marker and the end point of our trip, which is represented by a light green marker. Also we can see at which point of our route, we are commit a dangerous driving event. All dangerous driving points are represented with a red marker. When we tap in a dangerous driving point it is presented the name of the current event. In the figure below we can see an example of the map tab with the route and the markers as we mentioned before.

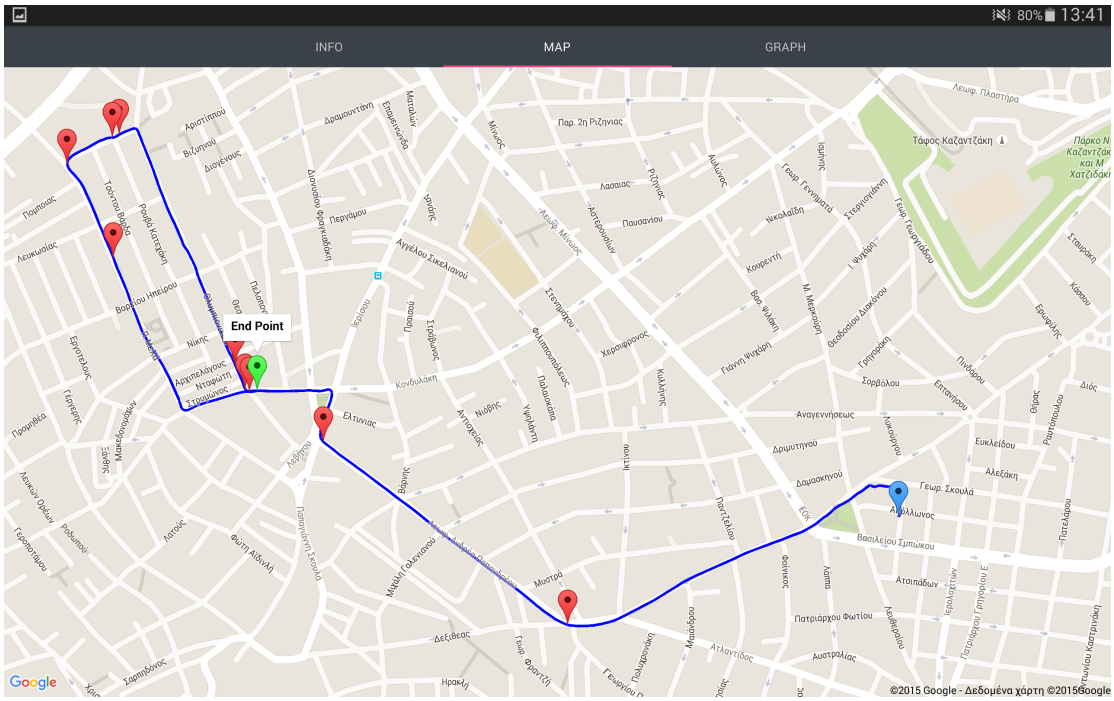


Figure 30 - Map Tab screen of our application.

In the last tab is presented the graph tab. In graph tab is presented 3 line charts, the acceleration line chart, the deceleration line chart and the turn line chart. We can see one of the 3 charts at the time. If we want to change graph chart we tap on the radio button of the chart we want under the displayed chart.

The x-axis represents the time in milliseconds and the y-axis represent the values of the detection method we have already chosen. These values can be can acceleration values (acceleration detection method – m/s^2) or orientation values (sensor fusion method – rad/s).

The charts are scrollable in the x-axis, which means that by scrolling horizontally we can see the acceleration or orientation values in relation with the time. The displayed x-axed duration is 60 seconds and as we scroll we can see the next 60 seconds.

Except the acceleration, deceleration and turn line charts we can see the thresholds lines for each chart. The thresholds line represented with a red direct lines with the threshold value for each event. If there is a value that has exceed the threshold line means than we have commit a dangerous driving event of the

current chart (acceleration, deceleration, turn) at this particular time. In the figures below we can see an example of these 3 line charts.

Figure 31 - Acceleration Line Chart of Graph Tab screen of our application.

Figure 32 - Deceleration Line Chart of Graph Tab screen of our application.

Figure 33 - Turn Line Chart of Graph Tab screen of our application.

7.1.5 - My Trips

In the main menu we choose the "My Trips" option when we want to browse previous trips. When the user chooses the "My Trips" option, all the recorded trips are presented in a scrollable list. Each trip is represented with the name "Trip " and an auto increment number, followed by a timestamp, which declares the date and the time that the trip begun. In the following figure we can see the list of the trips of a user.

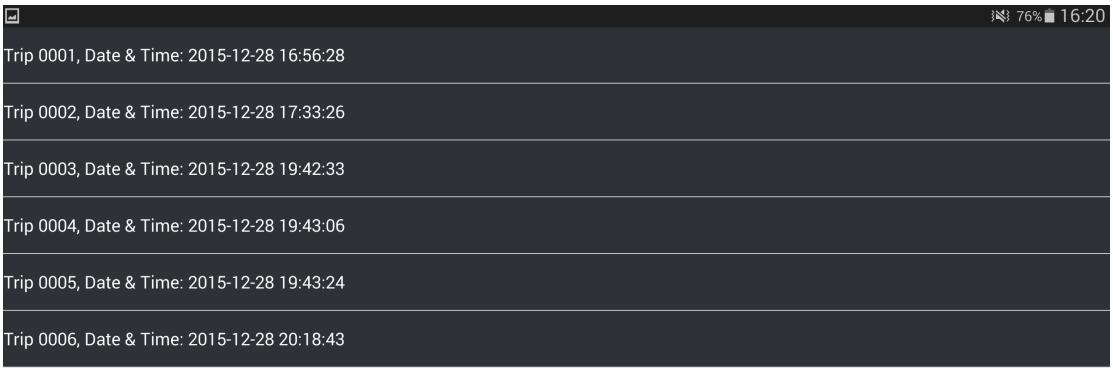


Figure 34 - "My Trip" screen. We can see the list of our trips.

7.1.5 - Settings

In the main menu there is also a "Settings" option when a user want to set his preferences. On of the main settings that the user can change is the standard monitor, which declares the preferred monitor style. The preferred monitor style can be the Basic Monitor and the Map Monitor. The other main setting is the choice of the detection method, which is the choice of the data, which the system will evaluate in order to detect safe or dangerous driving events and the driving behavior of the user. The detection method can be the accelerometer sensor or the sensor fusion method. Also from setting we can enable the GPS, if it's not enabled and to read more about the application. In the following figure we can see a snapshot of our settings.

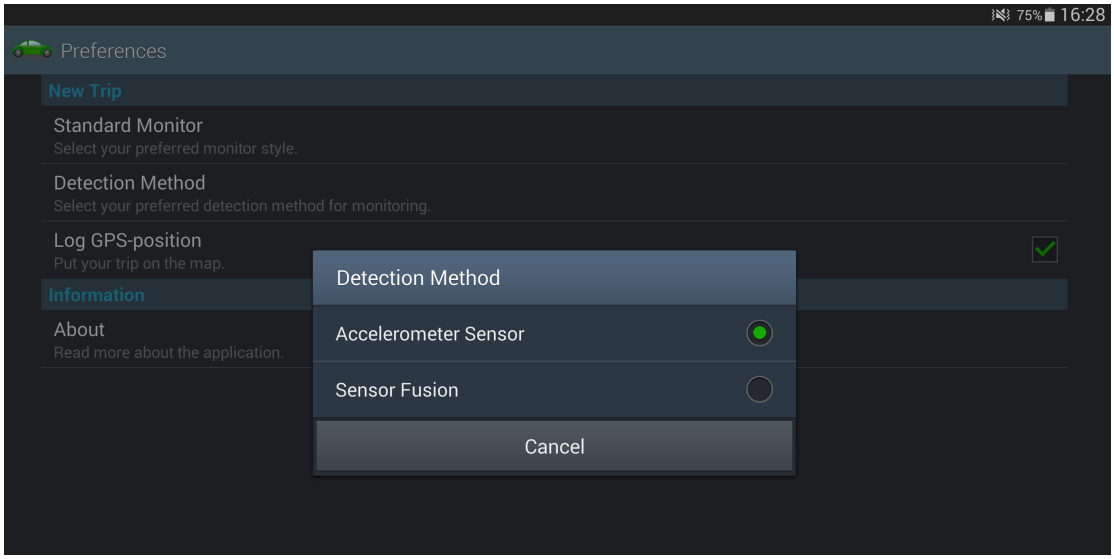


Figure 35 - Select detection method from the settings of our application.

7.2 E-Platform of Auto UBI System

We developed a web platform that can be used by a usage-based insurance company. The purpose of the platform is to check and evaluate the trip data of the company's drivers.

Via the particular platform an employee of the insurance company can have access to the list and data of all drivers of the company by entering the administrator's username and password. In the figure below is presented the log in system of the platform.

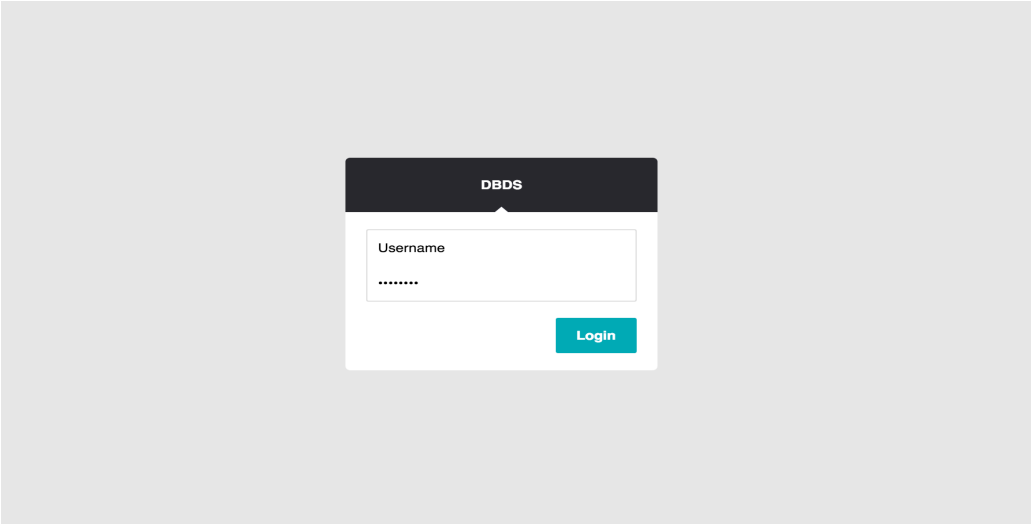


Figure 36 - Login Screen of UBI Portal.

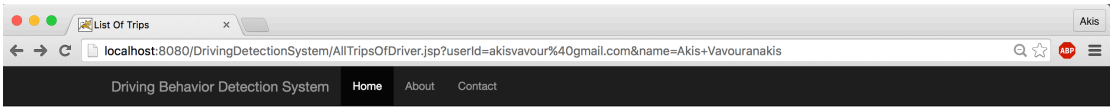
When an employee of the company logs in successfully to the system, can browse in a table of driver's data. In this table are presented all data of every driver. The presented data of the driver are: Full name, e-mail, Address, Phone Number and the license plate number of his vehicle. In the following figure we can see an example of the table with the data of every company's customer (driver).

List of Drivers

No	Name	Email	Address	Phone Number	License Plate Number	Display Trips
1	Mara Vavouranaki	mara@gmail.com	roufou 4	6875467432	HTK 2765	Select
2	Michael Vavouranakis	michael@gmail.com	dimokratias 3	6987427659	HMN 7898	Select
3	Athina Petromixelaki	athina@gmail.com	merkouri 8	6789457438	HNR 2356	Select
4	Akis Vavouranakis	akisvavour@gmail.com	apolonos 1	6944809166	HKT 2646	Select
5	Nikos Papazoglou	nikosp@yahoo.com	petlempouri 3	6987456328	HKT 5423	Select
6	Kostas Tsiclis	kostast@gmail.com	Papandreou 13	6754378934	YNE 5432	Select
7	Nikos Chairetis	nikosc@yahoo.com	merkouri 34	6545645345	HKT 5782	Select
8	Orestis Saridakis	orestis@gmail.com	kapadokias 35	6529835678	HTC 5439	Select
9	Stelios Andreadakis	steliosa@gmail.com	thessalonikis 2	6789345123	HMK 5421	Select

Figure 37 - List of all registered drivers in the program of UBI Company.

As we can see in the right side of every row (or driver) of the table, there is a "Select" button. When we want to browse the past trips of a particular driver, we click the "Select" button in the row of the particular driver. For example we click to browse the list of trips of a driver named "Akis Vavouranakis". In the figure below we present the list of trips of the driver named "Akis Vavouranakis".



List of Trips

Driver's Name: Akis Vavouranakis , Average Score: 9.705659/10 , Average Rating: Very Good

No	Timestamp	Distance	Duration	Average Speed	Detection Method	HAP*	HDP*	SLTP*	SRTP*	SLLCP*	SRLCP*	Score	Rating	View Map	View Graph
1	2015-12-30 15:37	0,50 km	1.51 min	16,13 km/h	ACC	5.88235 %	0 %	0 %	0 %	0 %	0 %	9.94118/10	Excellent	Select	Select
2	2015-12-30 15:41	0,88 km	2.40 min	19,78 km/h	ACC	0 %	0 %	14.2857 %	0 %	0 %	0 %	9.91667/10	Excellent	Select	Select
3	2015-12-30 15:45	2,12 km	6.40 min	19,06 km/h	ACC	0 %	7.14286 %	33.3333 %	0 %	0 %	0 %	9.77472/10	Excellent	Select	Select
4	2015-12-30 15:53	2,57 km	3.39 min	42,09 km/h	ACC	9.09091 %	0 %	16.6667 %	0 %	100 %	0 %	9.67832/10	Very Good	Select	Select
5	2015-12-30 15:57	3,14 km	9.57 min	18,91 km/h	ACC	0 %	0 %	28.5714 %	0 %	0 %	0 %	9.77778/10	Excellent	Select	Select
6	2015-12-30 16:08	1,50 km	2.16 min	39,60 km/h	ACC	16.6667 %	0 %	28.5714 %	100 %	0 %	0 %	9.45833/10	Very Good	Select	Select
7	2015-12-30 16:11	7,88 km	5.37 min	84,03 km/h	ACC	0 %	25 %	37.5 %	28.5714 %	100 %	0 %	9.33824/10	Very Good	Select	Select
8	2015-12-30 16:18	0,49 km	0.58 min	30,37 km/h	ACC	0 %	0 %	33.3333 %	0 %	0 %	0 %	9.75/10	Very Good	Select	Select
9	2015-12-30 16:20	1,18 km	5.59 min	11,77 km/h	ACC	11.7647 %	0 %	33.3333 %	0 %	0 %	0 %	9.71569/10	Very Good	Select	Select

*HAP=Hard Acceleration Penalty

Figure 38 - List of trips (and data of trips) of a particular driver.

In the list of trips, we can see the driving history of every driver. In the presented table, every row of the table represents a trip. Lots of data are presented for every trip. The presented general data are: The ***timestamp*** of the trip, which is the date and the starting time of the trip, the ***distance*** of the trip in km, the ***duration*** of the

trip, which is the actual duration of the trip (when the vehicle is in motion) and the ***average speed*** of the vehicle (km/h).

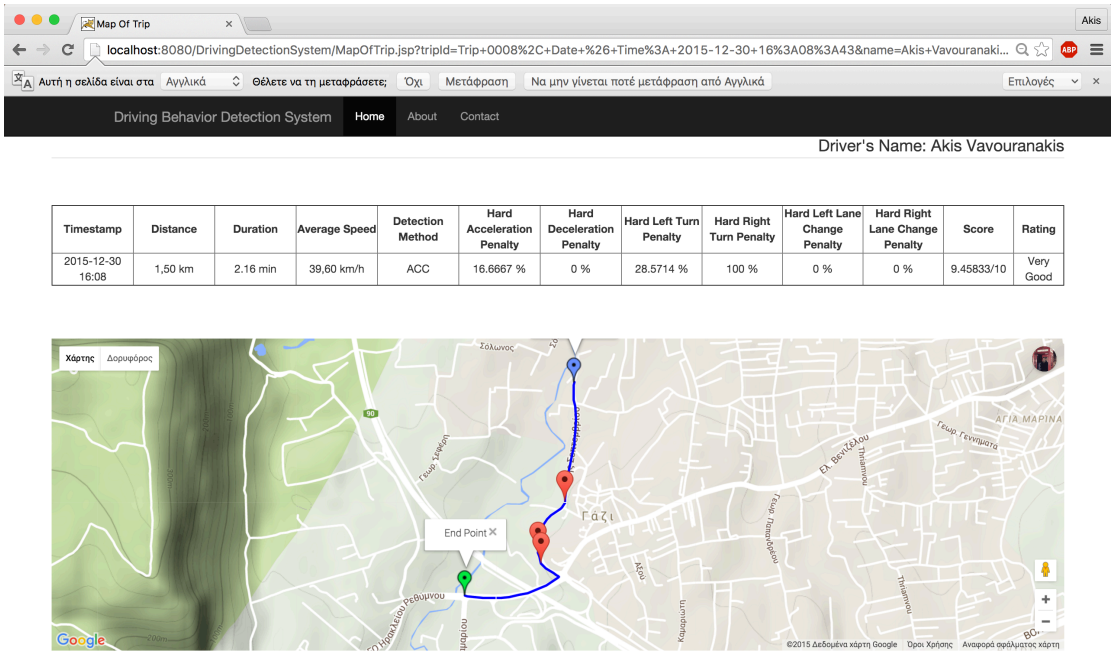
Also it is presented the selected from the user detection method (Accelerometer sensor data or Sensor fusion orientation data) and the percentage of penalties of all dangerous driving events. The percentages of penalties of the driving events that are presented are: ***Hard Acceleration Penalty*** (HAP), ***Hard Deceleration Penalty*** (HDP), ***Sharp Left Turn Penalty*** (SLTP), ***Sharp Right Turn Penalty*** (SRTP), ***Sharp Left Lane Change Penalty*** (SLLCP) and the ***Sharp Right Lane Change Penalty*** (SRLCP).

The ***score*** and the ***rating*** of every trip are also presented. The score of every trip is calculated based on the already above mentioned penalties. The max score for every trip is 10. Based on the score the rating (driver's behavior) of every trip is calculated. The rating can be: Very Bad, Bad, Good, Very Good or Excellent.

In the line above the array and its elements, we can see some average data of the driver. First of all, the name of the driver is displayed, which is followed by the ***average score*** and the ***average rating*** of all his trips.

As we can see, in the data table (which contains the list of trips of a particular user), in the last two columns of each row (trip) there are two "select" buttons. By selecting the first one, the route of the trip is displayed in a map and by selecting the second the data of the trip are presented in graphs. Also a table with the information of the particular trip is presented before the map.

In the following figure we can see how the route of the particular trip is displayed in a map when we select "View Map".



Timestamp	Distance	Duration	Average Speed	Detection Method	Hard Acceleration Penalty	Hard Deceleration Penalty	Hard Left Turn Penalty	Hard Right Turn Penalty	Hard Left Lane Change Penalty	Hard Right Lane Change Penalty	Score	Rating
2015-12-30 16:08	1,50 km	2.16 min	39,60 km/h	ACC	16.6667 %	0 %	28.5714 %	100 %	0 %	0 %	9.45833/10	Very Good

Figure 39 - Map Screen of our portal with the route and data of a particular trip.

We can see the route of our trip with a blue line. The starting point of our trip is displayed with a blue marker and the end of our trip is displayed with a light green marker. All dangerous driving events are displayed with a red marker. If we tap a red marker, the name of the dangerous driving event is displayed. The map is based on Google maps and we can zoom In/Out on it.

By selecting the "View Graph" select button, 3 line charts are presented, the acceleration line chart, the deceleration line chart and the turn line chart. The x-axis represents the time in milliseconds and the y-axis represent the values of the detection method we have already chosen. These values can be can acceleration values (acceleration detection method – m/s^2) or orientation values (sensor fusion method – rad/s).

Except the acceleration, deceleration and turn line charts we can see the thresholds lines for each chart. The thresholds line represented with a red direct lines with the threshold value for each event. If there is a value that has exceed the threshold line means than we have commit a dangerous driving event of the current chart (acceleration, deceleration, turn) at this particular time. Also a table with the information of the particular trip is presented before the graphs.

In the figures below we can see an example of these 3 line charts.

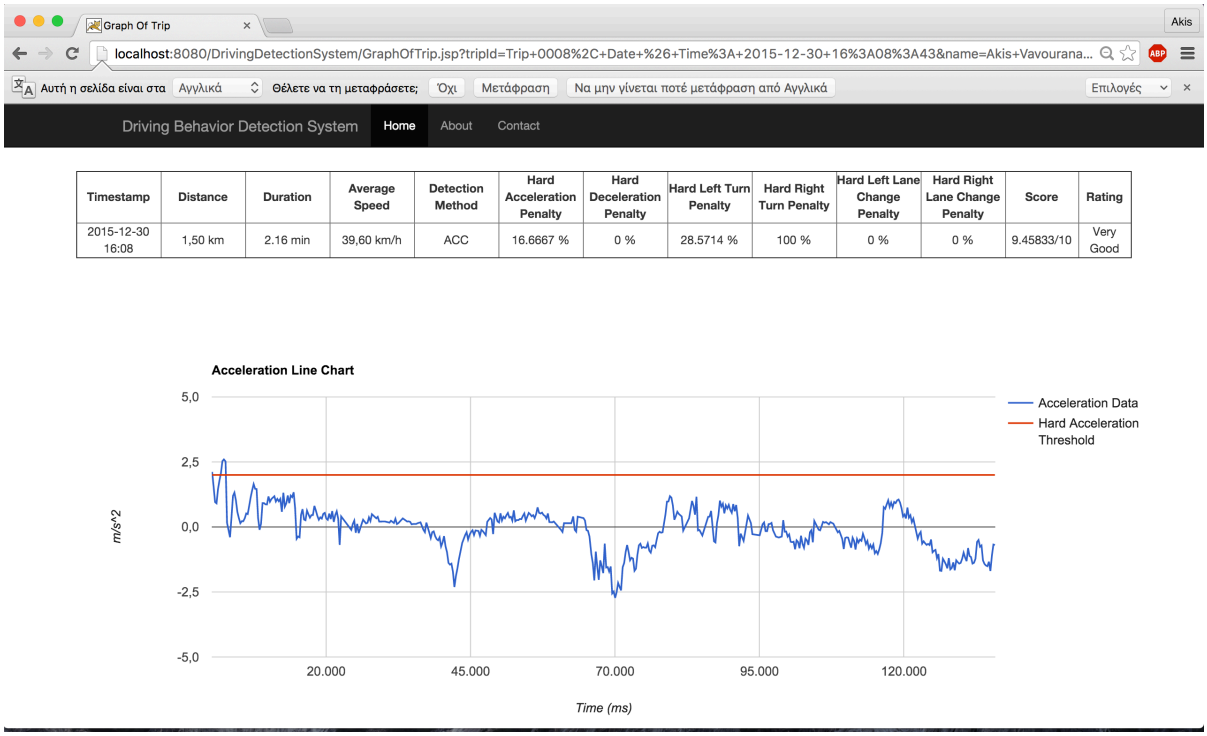


Figure 40 - Acceleration line chart with data of a particular trip.

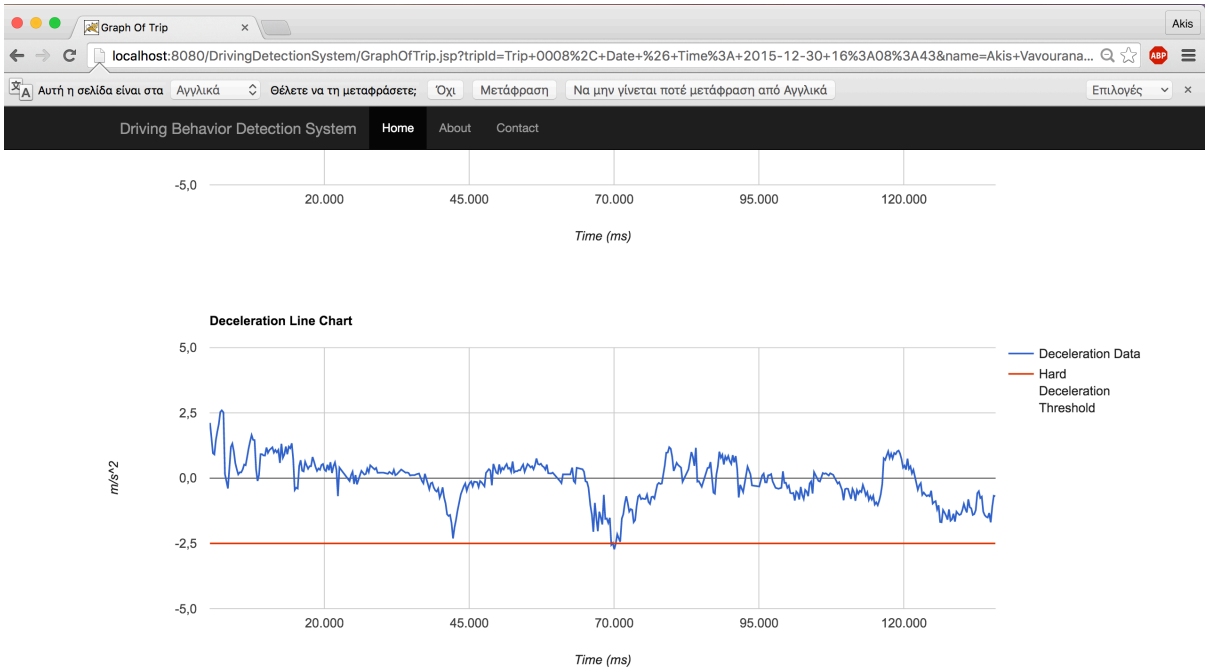


Figure 41 - Deceleration line chart of a particular trip.

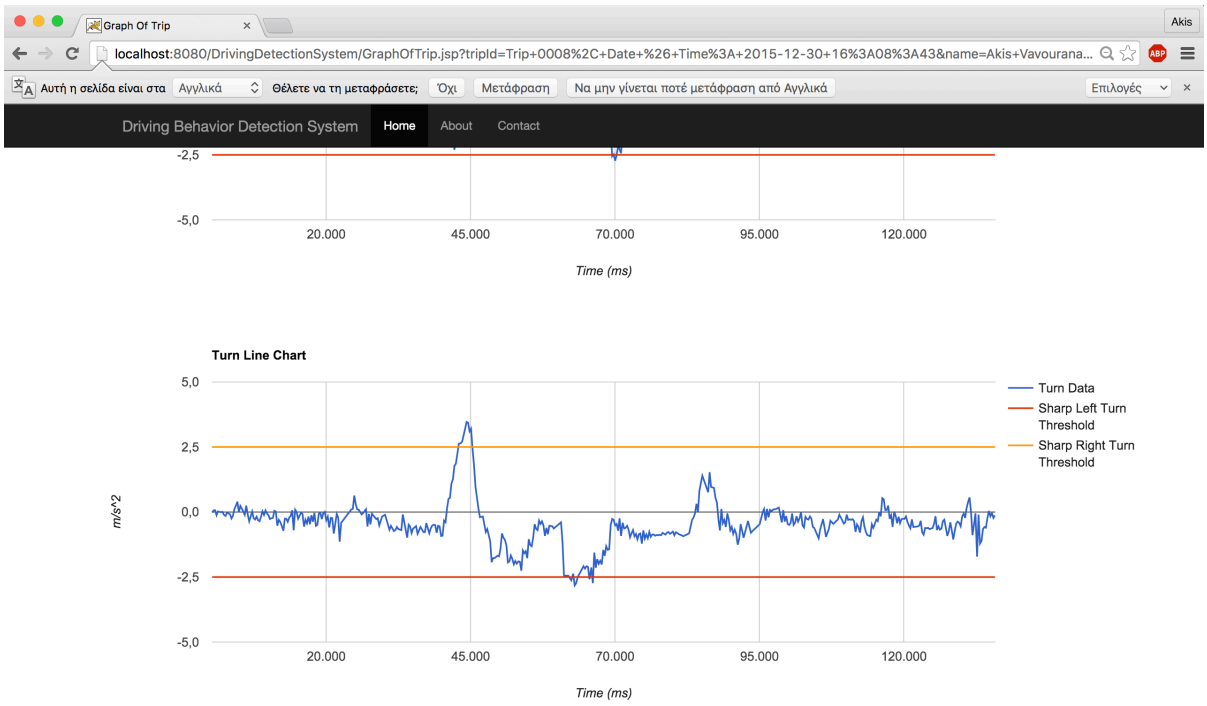


Figure 42 - Turn line chart of a particular trip.

Chapter 8 - Used Technologies & Tools

8.1 Android Application Development

For the implementation of our application we choose to develop a Native Application. A native mobile app is a smartphone application that is coded in a specific programming language, such as Objective C for iOS and Java for Android operating systems. Native mobile apps provide fast performance and a high degree of reliability. They also have access to a phone's various devices, such as its camera, sensors and address book.

We choose the Android platform, where the Java language is used, along with the Android Studio integrated development environment (IDE). This Android Studio is based on JetBrains' IntelliJ IDE and replaced Eclipse as the official development tool for Android applications. There are several steps to building an Android application. The steps include the following:

- Define the user interface. The UI for an application is generally defined as a series of layout files. These are XML-based files that describe the controls on a screen and the relationship of their layouts relative to one another.
- Add image assets, language translations and other resources. Android refers to non-code assets of a project as resources. These are placed in the project in a directory structure. At runtime, Android dynamically loads content from this directory structure.
- Write Java code to respond to various events that occur from the controls on a given screen and from changes in the lifecycle of an application. Java code is also responsible for loading the layout and menu files associated with each screen. And it's used to control the flow from one screen to the next.
- Export the completed Android application as a file that can be uploaded to Google Play or shared with others directly.

8.1.1 - Why Android?

We developed our application in the Android Platform because of 5 reasons. These 5 reasons are:

- **Portability**. Native Android apps are developed using the Java programming language, and can easily be ported to other mobile operating systems like Blackberry, Symbian and Ubuntu. In addition, Android apps can also be ported easily to Chrome OS. Not surprisingly, Microsoft has also announced that it will provide an easy method to port Android apps to Windows 10 devices.

- **Android Studio** is an excellent IDE, based on the equally excellent IntelliJ IDE. As the name suggests, Android Studio is an IDE designed and developed specifically for Android app development. It is blazingly fast and efficient, and you can setup a new Android project for different types of Android apps within seconds. When Android was launched, Android app development was done with Eclipse and the Android Developer Tools

plugin. However, that changed with the release of Android Studio. Some key features include: Gradle-based build system, Live-layout WYSIWYG Editor with real time app layout rendering, Option to preview a layout on multiple screen configurations while editing, Build variants and multiple apk file generation, Lint tools (used to catch usability, performance, version compatibility and other issues), Supports developing Android Wear, TV and Auto apps, Enables app integration with Google Cloud Platform (App Engine and Google Cloud Messaging)

- **Java** is a proven and powerful programming language, used on a wide range of devices and operating systems. Learning Java can open doors for other opportunities, including the ability to develop applications for other operating systems (Windows, Linux) and devices. Developing for iOS, on the other hand, requires that you learn one of Apple's development languages (Objective C or Swift). Both of these languages are really only used for Apple-centric development (iOS and OS X), and the skills needed to develop in these languages cannot be carried over to other operating systems. But to be fair, Apple has announced Swift will be open sourced, with Linux tools available before the end of the year.

- **Profitability**. The general consensus has always been that the iPhone is used by richer and more affluent users, and so, iPhone users are more likely to spend money on apps than Android users. This might have been true in the past, but not any more. In most app categories, Android apps have been found to be as profitable (even more profitable in some instances) as iPhone apps, both for initial app purchases and for in-app purchases. Also, with many apps using a free with ads model, as long as the ads are being shown to app users, the app generates income. According to DAU-UP, the average revenue per user for Android games was a measly 20% of that from iOS games in January 2014. By December 2014, the figure had spiked to 65%. In addition, advertising costs are generally lower on Android devices, which means that apps can advertise to more users on Android devices than users on iOS devices for the same amount.

- **Market share**. This has to be the number one reason why indie developers should develop for Android first. According to IDC, Android absolutely dominated the number of smartphones shipped worldwide in the first three months of 2015, with 78% market share. The estimated total number of Android devices in the hands of consumers, as at December 2014, according to statista.com, lies north of 1.6 billion. This is a staggering amount, and a very large potential market of users. Compare this with an estimated 395 million iOS devices, and 46 million and 45 million Windows and Blackberry devices respectively. If you are designing an app (or game) for the general public, it makes economic sense to target the platform that would give you the greatest access to potential users.

8.1.2 - Introduction to Android

Android [27] is a mobile operating system (OS) currently developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touchscreen devices, Google has further developed Android TV for televisions, Android Auto for cars, and Android Wear for wristwatches, each with a specialized user interface. Variants of Android are also used on notebooks, game consoles, digital cameras, and other electronics. As of 2015, Android has the largest installed base of all operating systems.

Initially developed by Android, Inc., which Google bought in 2005, Android was unveiled in 2007, along with the founding of the Open Handset Alliance – a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices. As of July 2013, the Google Play store has had over one million Android applications ("apps") published, and over 50 billion applications downloaded. An April–May 2013 survey of mobile application developers found that 71% of developers create applications for Android, and a 2015 survey found that 40% of full-time professional developers see Android as their priority target platform, which is comparable to Apple's iOS on 37% with both platforms far above others. At Google I/O 2014, the company revealed that there were over one billion active monthly Android users, up from 538 million in June 2013.

Android's source code is released by Google under open source licenses, although most Android devices ultimately ship with a combination of open source and proprietary software, including proprietary software required for accessing Google services. Android is popular with technology companies that require a ready-made, low-cost and customizable operating system for high-tech devices. Its open nature has encouraged a large community of developers and enthusiasts to use the open-source code as a foundation for community-driven projects, which add new features for advanced users or bring Android to devices originally shipped with other operating systems. At the same time, as Android has no centralized update system most Android devices fail to receive security updates: research in 2015 concluded that almost 90% of Android phones in use had known but unpatched security vulnerabilities due to lack of updates and support. The success of Android has made it a target for patent litigation as part of the so-called "smartphone wars" between technology companies.

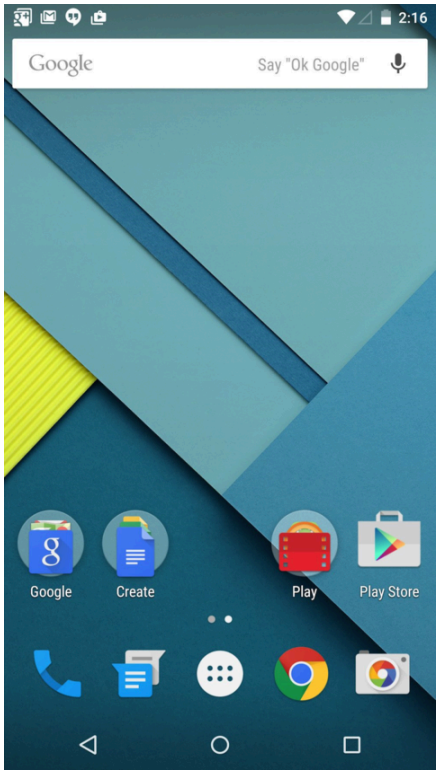


Figure 43 - Android 6.0 home screen.

Features

Interface. Android's default user interface is mainly based on direct manipulation, using touch inputs that loosely correspond to real-world actions, like swiping, tapping, pinching, and reverse pinching to manipulate on-screen objects, along with a virtual keyboard. Game controllers and full-size physical keyboards are supported via Bluetooth or USB. The response to user input is designed to be immediate and provides a fluid touch interface, often using the vibration capabilities of the device to provide haptic feedback to the user. Internal hardware, such as accelerometers, gyroscopes and proximity sensors are used by some applications to respond to additional user actions, for example adjusting the screen from portrait to landscape depending on how the device is oriented, or allowing the user to steer a vehicle in a racing game by rotating the device, simulating control of a steering wheel.

Android devices boot to the homescreen, the primary navigation and information "hub" on Android devices that is analogous to the desktop found on personal computers. (Android also runs on regular personal computers, as described below). Android homescreens are typically made up of app icons and widgets; app icons launch the associated app, whereas widgets display live, auto-updating content, such as the weather forecast, the user's email inbox, or a news ticker directly on the homescreen. A homescreen may be made up of several pages, between which the user can swipe back and forth, though Android's homescreen interface is heavily customisable, allowing users to adjust the look and feel of the devices to their tastes. Third-party apps available on Google Play and other app stores can extensively re-theme the homescreen, and even mimic the look of other

operating systems, such as Windows Phone. Most manufacturers, and some wireless carriers, customise the look and feel of their Android devices to differentiate themselves from their competitors. Applications that handle interactions with the homescreen are called "launchers" because they, among other purposes, launch the applications installed on a device.

Along the top of the screen is a status bar, showing information about the device and its connectivity. This status bar can be "pulled" down to reveal a notification screen where apps display important information or updates, such as a newly received email or SMS text, in a way that does not immediately interrupt or inconvenience the user. Notifications are persistent until read (by tapping, which opens the relevant app) or dismissed by sliding it off the screen. Beginning on Android 4.1, "expanded notifications" can display expanded details or additional functionality; for instance, a music player can display playback controls, and a "missed call" notification provides buttons for calling back or sending the caller an SMS message.

Android provides the ability to run applications that change the default launcher, and hence the appearance and externally visible behaviour of Android. These appearance changes include a multi-page dock or no dock, and many more changes to fundamental features of the user interface.

Applications. Applications ("apps"), which extend the functionality of devices, are written using the Android software development kit (SDK) and, often, the Java programming language that has complete access to the Android APIs. Java may be combined with C/C++, together with a choice of non-default runtimes that allow better C++ support; the Go programming language is also supported since its version 1.4, which can also be used exclusively although with a restricted set of Android APIs. The SDK includes a comprehensive set of development tools, including a debugger, software libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Initially, Google's supported integrated development environment (IDE) was Eclipse using the Android Development Tools (ADT) plugin; in December 2014, Google released Android Studio, based on IntelliJ IDEA, as its primary IDE for Android application development. Other development tools are available, including a native development kit (NDK) for applications or extensions in C or C++, Google App Inventor, a visual environment for novice programmers, and various cross platform mobile web applications frameworks. In January 2014, Google unveiled an framework based on Apache Cordova for porting Chrome HTML 5 web applications to Android, wrapped in a native application shell.

Android has a growing selection of third-party applications, which can be acquired by users by downloading and installing the application's APK (Android application package) file, or by downloading them using an application store program that allows users to install, update, and remove applications from their devices. Google Play Store is the primary application store installed on Android devices that comply with Google's compatibility requirements and license the Google Mobile Services software. Google Play Store allows users to browse, download and update applications published by Google and third-party

developers; as of July 2013, there are more than one million applications available for Android in Play Store. As of July 2013, 50 billion applications have been installed. Some carriers offer direct carrier billing for Google Play application purchases, where the cost of the application is added to the user's monthly bill.

Due to the open nature of Android, a number of third-party application marketplaces also exist for Android, either to provide a substitute for devices that are not allowed to ship with Google Play Store, provide applications that cannot be offered on Google Play Store due to policy violations, or for other reasons. Examples of these third-party stores have included the Amazon Appstore, GetJar, and SlideMe. F-Droid, another alternative marketplace, seeks to only provide applications that are distributed under free and open source licenses.

Memory Management. Since Android devices are usually battery-powered, Android is designed to manage memory (RAM) to keep power consumption at a minimum, in contrast to desktop operating systems which generally assume they are connected to unlimited mains electricity. When an Android application is no longer in use, the system will automatically suspend it in memory; while the application is still technically "open", suspended applications consume no resources (for example, battery power or processing power) and sit idly in the background until needed again. This brings a dual benefit by increasing the general responsiveness of Android devices, since applications do not need to be closed and reopened from scratch each time, and by ensuring that background applications do not consume power needlessly.

Android manages the applications stored in memory automatically: when memory is low, the system will begin killing applications and processes that have been inactive for a while, in reverse order since they were last used (oldest first). This process is designed to be invisible to the user, so that users do not need to manage memory or the killing of applications themselves. Lifehacker reported in 2011 that third-party task killers were doing more harm than good.

Development

Android is developed in private by Google until the latest changes and updates are ready to be released, at which point the source code is made available publicly. This source code will only run without modification on select devices, usually the Nexus series of devices. The source code is, in turn, adapted by OEMs to run on their hardware. Android's source code does not contain the often proprietary device drivers that are needed for certain hardware components.

In 2007, the green Android logo was designed for Google by a graphic designer Irina Blok. The design team was tasked with a project to create a universally identifiable icon with the specific inclusion of a robot in the final design. After numerous design developments based on science-fiction and space movies, the team eventually sought inspiration from the human symbol on restroom doors and modified the figure into a robot shape. As Android is open-sourced, it was agreed that the logo should be likewise, and since its launch the green logo has been reinterpreted into countless variations on the original design.

Update Schedule. Google provides major incremental upgrades to Android every six to nine months, with confectionery-themed names, which most devices are capable of receiving over the air. The latest major release is Android 6.0 "Marshmallow".

Compared to its primary rival mobile operating system, iOS, Android updates typically reach various devices with significant delays. For devices not under the Nexus brand, updates often arrive months from the time the given version is officially released, if at all. This is partly due to the extensive variation in hardware of Android devices, to which each upgrade must be specifically tailored, as the official Google source code only runs on their flagship Nexus devices. Porting Android to specific hardware is a time- and resource-consuming process for device manufacturers, who prioritize their newest devices and often leave older ones behind. Hence, older smartphones are frequently not updated if the manufacturer decides it is not worth their time, regardless of whether the phone is capable of running the update. This problem is compounded when manufacturers customize Android with their own interface and apps, which must be reapplied to each new release. Additional delays can be introduced by wireless carriers who, after receiving updates from manufacturers, further customize and brand Android to their needs and conduct extensive testing on their networks before sending the upgrade out to users.

The lack of after-sale support from manufacturers and carriers has been widely criticized by consumer groups and the technology media. Some commentators have noted that the industry has a financial incentive not to upgrade their devices, as the lack of updates for existing devices fuels the purchase of newer ones, an attitude described as "insulting". The Guardian has complained that the method of distribution for updates is complicated only because manufacturers and carriers have designed it that way. In 2011, Google partnered with a number of industry players to announce an "Android Update Alliance", pledging to deliver timely updates for every device for 18 months after its release; however, there has not been another official word about that alliance since its announcement.

In 2012, Google began decoupling certain aspects of the operating system (particularly core applications) so they could be updated through Google Play Store, independently of Android itself. One of these components, Google Play Services, is a closed-source system-level process providing APIs for Google services, installed automatically on nearly all devices running Android version 2.2 and higher. With these changes, Google can add new operating system functionality through Play Services and application updates without having to distribute an upgrade to the operating system itself. As a result, Android 4.2 and 4.3 contained relatively fewer user-facing changes, focusing more on minor changes and platform improvements.

Linux Kernel. Android's kernel is based on one of the Linux kernel's long-term support (LTS) branches. Since April 2014, Android devices mainly use versions 3.4 or 3.10 of the Linux kernel. The specific kernel version depends on the actual

Android device and its hardware platform; Android has used various kernel versions since the version 2.6.25 that was used in Android 1.0.

Android's variant of the Linux kernel has further architectural changes that are implemented by Google outside the typical Linux kernel development cycle, such as the inclusion of components like Binder, ashmem, pmem, logger, wakelocks, and different out-of-memory (OOM) handling. Certain features that Google contributed back to the Linux kernel, notably a power management feature called "wakelocks", were rejected by mainline kernel developers partly because they felt that Google did not show any intent to maintain its own code. Google announced in April 2010 that they would hire two employees to work with the Linux kernel community, but Greg Kroah-Hartman, the current Linux kernel maintainer for the stable branch, said in December 2010 that he was concerned that Google was no longer trying to get their code changes included in mainstream Linux. Some Google Android developers hinted that "the Android team was getting fed up with the process," because they were a small team and had more urgent work to do on Android.

In August 2011, Linus Torvalds said that "eventually Android and Linux would come back to a common kernel, but it will probably not be for four to five years". In December 2011, Greg Kroah-Hartman announced the start of Android Mainlining Project, which aims to put some Android drivers, patches and features back into the Linux kernel, starting in Linux 3.3. Linux included the autosleep and wakelocks capabilities in the 3.5 kernel, after many previous attempts at merger. The interfaces are the same but the upstream Linux implementation allows for two different suspend modes: to memory (the traditional suspend that Android uses), and to disk (hibernate, as it is known on the desktop). Google maintains a public code repository that contains their experimental work to re-base Android off the latest stable Linux versions.

The flash storage on Android devices is split into several partitions, such as /system for the operating system itself, and /data for user data and application installations. In contrast to desktop Linux distributions, Android device owners are not given root access to the operating system and sensitive partitions such as /system are read-only. However, root access can be obtained by exploiting security flaws in Android, which is used frequently by the open-source community to enhance the capabilities of their devices, but also by malicious parties to install viruses and malware.

Android is a Linux distribution according to the Linux Foundation, Google's open-source chief Chris DiBona, and several journalists. Others, such as Google engineer Patrick Brady, say that Android is not Linux in the traditional Unix-like Linux distribution sense; Android does not include the GNU C Library (it uses Bionic as an alternative C library) and some of other components typically found in Linux distributions.

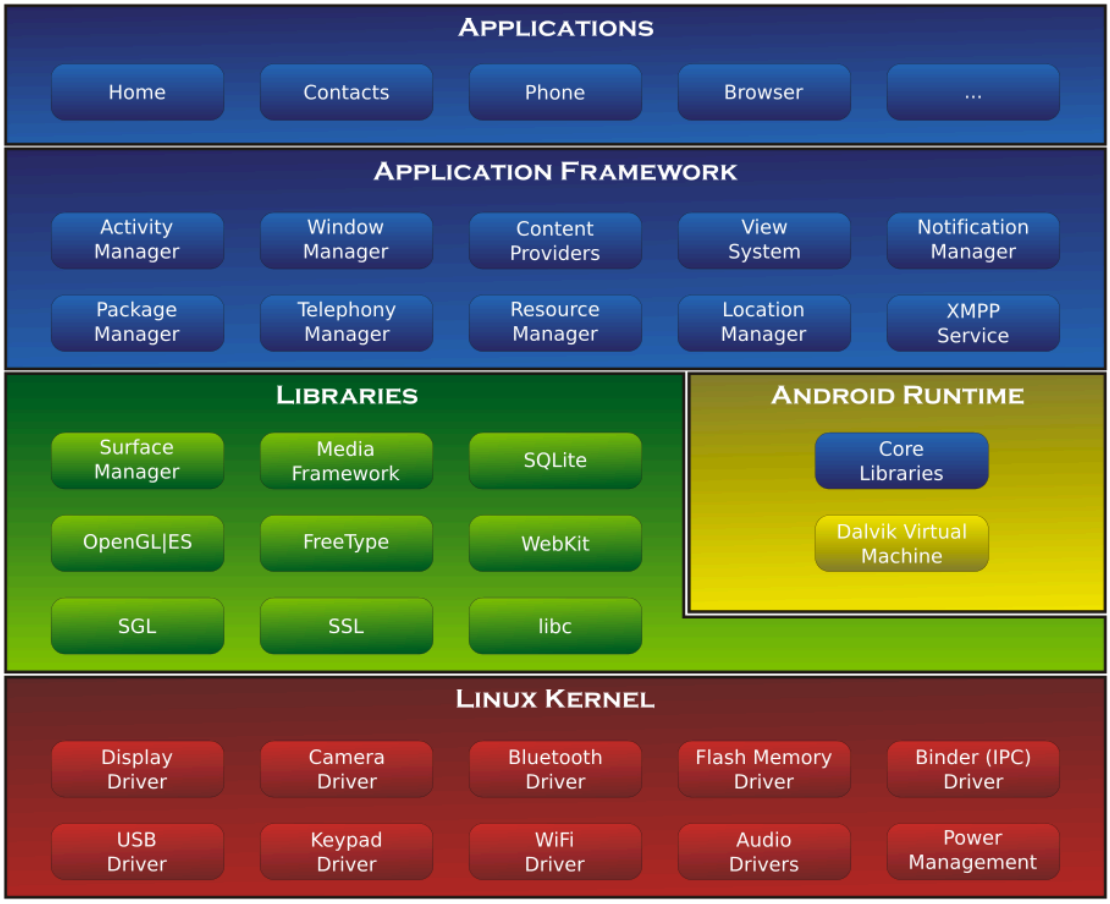


Figure 44 - Android's architecture diagram.

Software Stack. On top of the Linux kernel, there are the middleware, libraries and APIs written in C, and application software running on an application framework which includes Java-compatible libraries based on Apache Harmony. Development of the Linux kernel continues independently of other Android's source code bases.

Until version 5.0, Android used Dalvik as a process virtual machine with trace-based just-in-time (JIT) compilation to run Dalvik "dex-code" (Dalvik Executable), which is usually translated from the Java bytecode. Following the trace-based JIT principle, in addition to interpreting the majority of application code, Dalvik performs the compilation and native execution of select frequently executed code segments ("traces") each time an application is launched. Android 4.4 introduced Android Runtime (ART) as a new runtime environment, which uses ahead-of-time (AOT) compilation to entirely compile the application bytecode into machine code upon the installation of an application. In Android 4.4, ART was an experimental feature and not enabled by default; it became the only runtime option in the next major version of Android, 5.0.

In December 2015, Google announced that the next version of Android would switch to a Java implementation based on OpenJDK.

Android's standard C library, Bionic, was developed by Google specifically for Android, as a derivation of the BSD's standard C library code. Bionic itself has been designed with several major features specific to the Linux kernel. The main

benefits of using Bionic instead of the GNU C Library (glibc) or uClibc are its smaller runtime footprint, and optimization for low-frequency CPUs. At the same time, Bionic is licensed under the terms of the BSD licence, which Google finds more suitable for the Android's overall licensing model.

Aiming for a different licensing model, toward the end of 2012 Google switched the Bluetooth stack in Android from the GPL-licensed BlueZ to the Apache-licensed BlueDroid.

Android does not have a native X Window System by default, nor does it support the full set of standard GNU libraries. This made it difficult to port existing Linux applications or libraries to Android, until version r5 of the Android Native Development Kit brought support for applications written completely in C or C++. Libraries written in C may also be used in applications by injection of a small shim and usage of the JNI.

Open Source Community. Android has an active community of developers and enthusiasts who use the Android Open Source Project (AOSP) source code to develop and distribute their own modified versions of the operating system. These community-developed releases often bring new features and updates to devices faster than through the official manufacturer/carrier channels, with a comparable level of quality; provide continued support for older devices that no longer receive official updates; or bring Android to devices that were officially released running other operating systems, such as the HP TouchPad. Community releases often come pre-rooted and contain modifications not provided by the original vendor, such as the ability to overclock or over/undervolt the device's processor. CyanogenMod is the most widely used community firmware, and acts as a foundation for numerous others. There have also been attempts with varying degrees of success to port Android to iPhones, notably the iDroid Project.

Historically, device manufacturers and mobile carriers have typically been unsupportive of third-party firmware development. Manufacturers express concern about improper functioning of devices running unofficial software and the support costs resulting from this. Moreover, modified firmwares such as CyanogenMod sometimes offer features, such as tethering, for which carriers would otherwise charge a premium. As a result, technical obstacles including locked bootloaders and restricted access to root permissions are common in many devices. However, as community-developed software has grown more popular, and following a statement by the Librarian of Congress in the United States that permits the "jailbreaking" of mobile devices, manufacturers and carriers have softened their position regarding third party development, with some, including HTC, Motorola, Samsung and Sony, providing support and encouraging development. As a result of this, over time the need to circumvent hardware restrictions to install unofficial firmware has lessened as an increasing number of devices are shipped with unlocked or unlockable bootloaders, similar to Nexus series of phones, although usually requiring that users waive their devices' warranties to do so. However, despite manufacturer acceptance, some carriers in the US still require that phones are locked down, frustrating developers and customers.

8.1.3 - SQLite Database

All data are saved locally in our device, with the help of SQLite [28]. SQLite is a relational database management system contained in a C programming library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program.

SQLite is ACID-compliant and implements most of the SQL standard, using a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity.

SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems, among others. SQLite has bindings to many programming languages.

8.1.4 - Google Maps Android API

We display the routes and the dangerous driving events of our trips using Google Maps Android API [29]. This API allow to our users to explore the world with rich maps provided by Google. Identify locations with custom markers, augment the map data with image overlays, embed one or more maps as fragments, and much more.

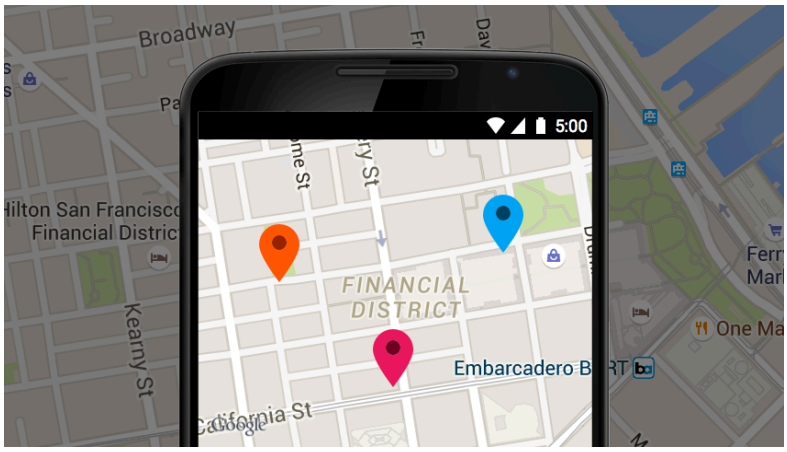


Figure 45 - Google Maps in Android device.

The Google Maps Android API allows you to include maps and customized mapping information in your app.

Add maps to your app

With Google Maps Android API v2, you can embed maps into an activity as a fragment with a simple XML snippet. The new Maps offer exciting features such as 3D maps; indoor, satellite, terrain, and hybrid maps; vector-based tiles for efficient caching and drawing; animated transitions; and much more.

Customize the map

Add markers onto the map to indicate special points of interest for your users. You can define custom colors or icons for your map markers to match your app's look and feel. To further enhance the app, draw polylines and polygons to indicate paths or regions, or provide complete image overlays.

Control the user's view

Give your users a different view of the world with the ability to control the rotation, tilt, zoom, and pan properties of the "camera" perspective of the map.

Add Street View to your app

Embed Street View into an activity and let your users explore the world through panoramic 360-degree views. Programmatically control the zoom and orientation (tilt and bearing) of the Street View camera, and animate the camera movements over a given duration.

8.1.5 - Graph View

In our app we create our line charts with the help of Graph View. GraphView [30] is a library for Android to programmatically create flexible and nice-looking diagrams. It is easy to understand, to integrate and to customize. Create Line Graphs, Bar Graphs, Point Graphs or implement your own custom types.

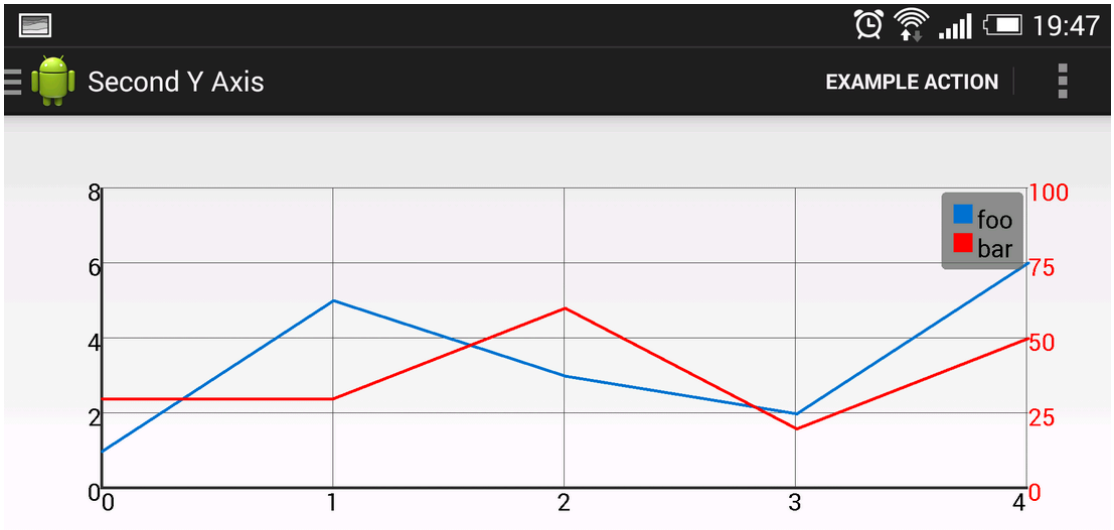


Figure 46 - Line graph with two y-scales.

8.2 Web Application Development

For the implementation of our web application we choose the Apache Tomcat Web Server [31]. Apache Tomcat, often referred to as Tomcat, is an open-source web server developed by the Apache Software Foundation (ASF). Tomcat implements several Java EE specifications including Java Servlet, JavaServer Pages (JSP), Java EL, and WebSocket, and provides a "pure Java" HTTP web server environment for Java code to run in.

Tomcat is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation, released under the Apache License 2.0 license, and is open-source software.

We send our JSON data to our server via http methods. JSON, is an open standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs. It is the primary data format used for asynchronous browser/server communication, largely replacing XML (used by AJAX).

The server receives and analyzes JSON data. After analyzing the data we store them to our MySQL database. JSP have access to MySQL database using JDBC. **Java Database Connectivity** (JDBC) is an application-programming interface (API) for the programming language Java, that defines how a client may access a database. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database, and is oriented towards relational databases. A JDBC-to-ODBC bridge enables connections to any ODBC-accessible data source in the Java virtual machine (JVM) host environment.

8.2.1 - JSP & Servlets

JavaServer Pages (JSPs) [32] are a specification for combining Java with HTML to provide dynamic content for Web pages. When you create dynamic content, JSPs are more convenient to write than HTTP servlets because they allow you to embed Java code directly into your HTML pages, in contrast with HTTP servlets, in which you embed HTML inside Java code.

JSPs are Web pages coded with an extended HTML that makes it possible to embed Java code in a Web page. JSPs can call custom Java classes, called taglibs, using HTML-like tags. The WebLogic appc compiler weblogic.appc generates JSPs and validates descriptors. You can also precompile JSPs into the WEB-INF/classes/ directory or as a JAR file under WEB-INF/lib/ and package the servlet class in the Web archive to avoid compiling in the server. Servlets and JSPs may require additional helper classes to be deployed with the Web application.

JSPs enable you to separate the dynamic content of a Web page from its presentation. It caters to two different types of developers: HTML developers, who are responsible for the graphical design of the page, and Java developers, who handle the development of software to create the dynamic content.

A **Servlet** [33] is a Java class that runs in a Java-enabled server. An HTTP servlet is a special type of servlet that handles an HTTP request and provides an HTTP response, usually in the form of an HTML page. The most common use of WebLogic HTTP servlets is to create interactive applications using standard Web browsers for the client-side presentation while WebLogic Server handles the business logic as a server-side process. WebLogic HTTP servlets can access databases, Enterprise JavaBeans, messaging APIs, HTTP sessions, and other facilities of WebLogic Server.

8.2.1 - MySQL

MySQL [34] is an open-source relational database management system (RDBMS); in July 2013, it was the world's second most[a] widely used RDBMS, and the most widely used open-source client–server model RDBMS. It is named after co-founder Michael Widenius's daughter, My. The SQL acronym stands for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. For proprietary use, several paid editions are available, and offer additional functionality.

MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack (and other "AMP" stacks). LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python." Free-software-open source projects that require a full-featured database management system often use MySQL. Applications that use the MySQL database include: TYPO3, MODx, Joomla, WordPress, phpBB, MyBB, Drupal and other software. MySQL is also used in many high-profile, large-scale websites, including Google (though not for searches), Facebook, Twitter, Flickr, and YouTube.

On all platforms except Windows, MySQL ships with no GUI tools to administer MySQL databases or manage data contained within the databases. Users may use the included command line tools, or install MySQL Workbench via a separate download. Many third party GUI tools are also available.

8.2.1 - Google Maps Web API

We display the routes and the dangerous driving events of our trips using Google Maps JavaScript API. The Google Maps JavaScript API [35] is a powerful, popular mapping API. It's simple to use to add maps to your website, or web or mobile application, and provides a wide range of services and utilities for data visualization, map manipulation, directions, and more.

8.2.1 - Google Charts

Google Charts [36] provides a perfect way to visualize data on your website. From simple line charts to complex hierarchical tree maps, the chart gallery provides a large number of ready-to-use chart types.

The most common way to use Google Charts is with simple JavaScript that you embed in your web page. You load some Google Chart libraries, list the data to be charted, select options to customize your chart, and finally create a chart object with an id that you choose. Then, later in the web page, you create a with that id to display the Google Chart.

Charts are exposed as JavaScript classes, and Google Charts provides many chart types for you to use. The default appearance will usually be all you need, and you can always customize a chart to fit the look and feel of your website. Charts are highly interactive and expose events that let you connect them to create complex dashboards or other experiences integrated with your webpage. Charts are

rendered using HTML5/SVG technology to provide cross-browser compatibility (including VML for older IE versions) and cross platform portability to iPhones, iPads and Android. Your users will never have to mess with plugins or any software. If they have a web browser, they can see your charts.

All chart types are populated with data using the DataTable class, making it easy to switch between chart types as you experiment to find the ideal appearance. The DataTable provides methods for sorting, modifying, and filtering data, and can be populated directly from your web page, a database, or any data provider supporting the Chart Tools Datasource protocol. (That protocol includes a SQL-like query language and is implemented by Google Spreadsheets, Google Fusion Tables, and third party data providers such as SalesForce. You can even implement the protocol on your own website and become a data provider for other services.)

Chapter 9 – Conclusion

9.1 Thesis Summary

This work shows that the user (driver) can benefit from a Usage-Based system; the system rewards the user for being a safe driver and also provides him with feedback on his driving habits, making him a better and safer driver. The device measures various elements of user's driving behavior, safe or hard breaking, safe or sharp accelerating, safe or sharp turns and safe or sharp lane changes. Also measures, the start time of trip, the duration of trip and the total travelled distance of trip. Also for every trip, the user gets a score and a rating, depending on how well he droves. The insurance company has access to all these data (trips data, routes and graphs) of his drivers via the e-platform.

Also in our work, we show that the driving behavior of the user could be estimated based on acceleration data of the accelerometer sensor or based on orientation data of sensor fusion method. Sensor fusion method combines data from the accelerometer, geomagnetic field sensor and the gyroscope. The driver can choose the detection method from the settings of the application's main menu. We propose to drivers to use accelerometer data as their detection method. The reason is not that the sensor fusion data are not accurate or reliable data but the sensor fusion data are not the best option for the detection of sharp turns or sharp lane changes. When we use only the accelerometer, the algorithm detects sharp turns or sharp lane changes faster than when we use the orientation data of sensor fusion method. From the other hand when we use the sensor fusion as detection method the system detects faster the safe/hard acceleration or deceleration of the vehicle than when we use only the acceleration data.

Using the sensor fusion method or accelerometer method it is understood that safety comes first and the system takes into account that sometimes hard deceleration and rapid accelerating or other dangerous driving events are necessary to avoid a collision. The system works to identify a pattern in your driving habits so the occasional hard brake will not have a significant (if any) impact on your potential rating. The discount of the drivers is depending on his average rating of his trips. So better rating means highest discount for his insurance.

By using the application and the e-platform UBI information system, the drivers and the insurance company have many benefits. Some of them are:

- Social and environmental benefits from more responsible and less unnecessary driving.
- Commercial benefits to the insurance company from better alignment of insurance with actual risk. Improved customer segmentation.
- Potential cost-savings for responsible customers.
- Technology that powers UBI enables other vehicle-to-infrastructure solutions including drive-through payments, emergency road assistance, etc.
- More choice for consumers on type of car insurance available to buy.

- Social benefits from accessibility to affordable insurance for young drivers - rather than paying for irresponsible peers, with this type of insurance young drivers pay for how they drive.
- Higher-risk drivers pay most per use, thus have highest incentive to change driving patterns or get off the roads, leaving roads safer.
- For telematics usage-based insurance: Continuous tracking of vehicle location enhances both personal security and vehicle security. The GPS technology could be used to trace the vehicle whereabouts following an accident, breakdown or theft.
- The same GPS technology can often be used to provide other (non insurance) benefits to consumers, e.g. satellite navigation.
- Gamification of the data encourages good driver behavior by comparison with other drivers.

Our system can serve some useful purposes but it has some limitations and drawbacks too. Some of them are:

- They system cannot detect the backward movement of the vehicle. During the user drives backwards, the detected driving events will be wrong. This wrong driving events will affect the score and the may the rating of the user.
- A limitation of our system has to do with the calibration process of the device. Before we start driving the application will tell us to follow some instructions for the calibration of the device. This process has to be repeated for each new trip and the whole process takes about 5 seconds to complete.
- Another limitation is when we start the calibration process when the car is on a slope. If the car is on a slope during the calibration procedure the reading data will be affected and this will lead to poor results.
- The device has to be in a fixed position during the trip. After the calibration and while monitoring you cannot move the device from its fixed position. If the device moved while monitoring the sensor's output data will be wrong and the evaluation of the data will not be accurate. In this case the user has to repeat the calibration procedure.
- Some of low cost Android devices have low quality sensors or processing power. So the readings of the sensor's data are not so accurate and the application slow down due to low processing power.
- All Android devices have in-built the most used sensors like accelerometer and gyroscope. The magnetometer sensor on the other side is not included in all smartphone devices. So without the magnetometer sensor the user cannot use the Sensor Fusion detection method.

9.2 Recommendation for Future Work

The research that has been undertaken for this thesis has highlighted a number of features on which further implementation would be beneficial. Some future features and implementations for this project are:

- As the driving style in this thesis was based mostly on the lateral and

longitudinal forces acting on a smartphone (vehicle), in a future work the phone camera could be used to detect distance between the vehicles or the car's position in the lane.

- Also another camera feature could also be implemented. It could take pictures based on distance or time. When the user views a previous trip, the pictures could be linked to the map. If there are some locations (where bad driving events occurred), the user could view the associated image to better understand the evaluation of the dangerous driving events. There could also be a record video feature that filmed the whole trip. These features would of course require the user to mount the phone in a way that guaranteed the camera free sight.
- The speed data acquired from the GPS or computed from the acceleration sensor could be used in combination with the location data from the GPS to detect speeding in various areas.
- It has to be mentioned that the vehicle, the mobile device and the nature of the road affect the characteristics of the sensor data. Due to this variation in characteristic, the accuracy of the system with fixed thresholds would be lower when tested under different conditions [37]. Therefore, the use of dynamic time warping for the different events would provide more accurate and reliable results.
- With the collected data, we could generate a map (map with dangerous areas) and see if there are some areas where most drivers had the same bad driving events.
- Another thing we could do with a database is for the users to upload their score, and generate a scoreboard. It could be a scoreboard for specific routes or areas. We could also make a user ranking system; to display each user's overall ranking and make it easier for friends to compare each other.
- Also another nice feature would be to make it possible to compare different trips on the phone. E.g. if someone drives the same way to work every day, it might be interesting to compare how they drove on different days.
- Another future development is to make our application social, with hopes of reaching more potential users for the insurance company. We will integrate our application with Facebook and Twitter, where the drivers could share their scores and routes.

Bibliography

[1] Usage based insurance, https://en.wikipedia.org/wiki/Usage-based_insurance

[2] "Usage-Based Insurance and Telematics". National Association of Insurance Commissioners. Retrieved 22 February 2014.

[3] J. Paefgen, T. Staake & F. Thiesse, "Resolving the Misalignment between Consumer Privacy Concerns and Ubiquitous IS Design: The Case of Usage-based Insurance", International Conference on Information Systems (ICIS), 2012

[4] Progressive Casualty Insurance Company, www.progressive.com

[5] Accelerometer, http://en.wikipedia.org/wiki/Accelerometer

[6] Gyroscope, http://en.wikipedia.org/wiki/Gyroscope

[7] Magnetometer, http://en.wikipedia.org/wiki/Magnetometer

[8] GPS, http://en.wikipedia.org/wiki/GPS_navigation_device

[9] Singh, P., Juneja, N., Kapoor, S.: Using mobile phone sensors to detect driving behavior. In: Proceedings of the 3rd ACM Symposium on Computing for Development, ACM (2013)

[10] Fazeen, M., Gozick, B., Dantu, R., Bhukhiya, M., Gonzalez, M.C.: Safe Driving Using Mobile Phones. In: IEEE Transactions on Intelligent Transportation Systems (2012)

[11] Chigurupa, S., Polavarap, S., Kancherla,Y., Nikhath, K.A.:Integrated Computing System for measuring Driver Safety Index. In: International Journal of Emerging Technology and Advanced Engineering, ISSN 2250-2459, Volume 2 (2012)

[12] Dai, J., Tang, J., Bai, X., Shen, Z., Xuan, D.:Mobile phone based drunk driving detection. In: Proc. 4th Int. Conf. Pervasive Health NO PERMISSIONS,, pp. 18 (2010)

[13] Johnson, D.A., Trivedi, M.M.:Driving Style Recognition using a smartphone as a sensor platform. In: IEEE 14th International Conference on Intelligent Transportation system, October (2011)

[14] H.Eren, S.Makinist, E.Akin, A.Yilmaz: Estimating Driving Behavior by a smartphone . In: Intelligent Vehicles Symposium, Alcala de Henares, Span, June(2012)

[15] Chalermporl Saiprasent and Wasan Pattara-Atikom: Smartphone Enabled Dangeroys Driving Report System. In: 46th Hawaii International Conference on System Sciences (2013)

[16] Chuang-Wen You, Martga Montes-de-Oca, Thomas J.Bao, Nicholas D. Lane, Hong Lu, Giuseppe Cardone, Lorenzo Torresani, Andrew T. Campbell: CarSafe: A Driver Safety App thath Detects Dangerous Driving Behavior using Dual – Cameras on Smartphones. In: UbiComp12, Pittsburg, USA, September(2012)

[17] Fadi Aloul, Imran Zualkernan, Ruba Abu-Salma, Humaid Al-Ali, May Al-Merri: iBump:Smartphone Application to Detect Car Accidents. In: IAICT, Bali 28-30 August 2014

[18] Nidhi Kalra, Gunjan Chugh, Divya Bansal : Analyzing Driving and Road Events via Smartphone. In: International Journal Of Computer Applications No.12, July 2014

[19] Jin-Hyuk Hong, Ben Margines, Anind K. Dey: A smartphone-based sensing platform to Model Aggressive Driving Behaviors. In: CHI , Toronto, Canada (2014)

[20] Johannes Paefgen, Flavius Kehr, Yudan Zhai, Florian Michahelles: Driving Behavior Analysis with Smartphones:Insights from a controlled Field Study. In: MUM'12, ULM, Germany (2012)

[21] Fr. Hørtvedt, Fr. Kvitvik, and J. A. Myrland. DriSMo - the driving quality application. Bachelor thesis, Gjøvik University College, May 2011.

[22] Atan2, https://en.wikipedia.org/wiki/Atan2

[23] Radoslav Stoichkov, Android Smartphone Application for Driving Style Recognition, Department of Electrical Engineering and Information Technology Institute for Media Technology, July 2013.

[24] P. Lawitzki. Application of Dynamic Binaural Signals in Acoustic Games. Master's thesis, Hochschule der Medien Stuttgart, 2012.

[25] P. Lawitzki, Android Sensor Fusion Tutorial. http://plaw.info/2012/03/android-sensor-fusion-tutorial/

[26] Exponential Moving Average, https://en.wikipedia.org/wiki/Moving_average

[27] Android (Operating System), https://en.wikipedia.org/wiki/Android_(operating_system)

[28] SQLite Database, https://en.wikipedia.org/wiki/SQLite

[29] Google Maps Android API,
https://developers.google.com/maps/documentation/android-api/

[30] GraphView - open source graph plotting library for Android
http://www.android-graphview.org/

[31] Apache Tomcat, http://tomcat.apache.org/

[32] Java Server Pages, https://en.wikipedia.org/wiki/JavaServer_Pages

[33] Java servlet, https://en.wikipedia.org/wiki/Java_servlet

[34] MySQL, https://en.wikipedia.org/wiki/MySQL

[35] Google Maps JavaScript API
https://developers.google.com/maps/documentation/javascript/

[36] Google Charts, https://developers.google.com/chart/

[37] R. Bhoraskar, N. Vankadhara, B. Raman, and P. Kulkarni. Wolverine: Traffic and road condition estimation using smartphone sensors. In 2012 Fourth International Conference on Communication Systems and Networks (COMSNETS), pages 1–6, 2012.

[38] Mrinal Haloi, Dinesh Babu Jayagopi: Characterizing driving behavior using automatic visual analysis. Proceedings of the 6th IBM Collaborative Academia Research Exchange Conference (I-CARE) on I-CARE 2014, pages 1-4.

[39] Z. Chen, J. Yu, Y. Zhu, Y. Chen and M. Li: D^3: Abnormal Driving Behaviors Detection and Identification Using Smartphone Sensors, 12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), 2015.

[40] A. Ashutosh, B. Piyush.K.Ingole: Smartphone based approach to monitor driving behavior and sharing of statistic, International Journal of Advanced Technology & Engineering Research (IJATER), 2nd International e-Conference on Emerging Trends in Technology (E-ICETT) 2014.

[41] P. Dhar, S. Shinde, N. Jadav, A. Bhaduri: Unsafe Driving Detection System using Smartphone as Sensor Platform, International Journal of Enhanced Research in Management & Computer Applications, ISSN: 2319-7471 Vol. 3 Issue 3, March-2014, pp: (65-70).

[42] El Hosin Gazali, Monitoring Erratic Driving Behavior caused by Vehicle Overtaking using Off-the-shelf Technologies, Master of Science in Computer Science, University of Dublin, October 2010.

[43] Ming Liu, A study of Mobile Sensing Using Smartphones, Hindawi Publishing Corporation International Journal of Distributed Sensor Networks Volume 2013, Article ID 272916, 11 pages.

[44] Priyanka B. Shinde1, Vikram A. Mane2 P.G. Student (E&TC), Context aware driver's behavior detection system using Zigbee: Result, Annasaheb Dange College of Engineering & Technology Ashta, India1 International Journal of innovative research in electrical, electronics, instrumentation and control engineering Vol. 3, Issue 1, January 2015.

[45] P. Tharangai Thamil, S. Vanitha: Survey on Rash Driving Detection Using Acceleration and Orientation Sensors, International Journal of Scientific Research Engineering & Technology (IJSRET), ISSN 2278 – 0882 Volume 4, Issue 3, March 2015

[46] V. Corcoba Magana, M. Munoz-Organero. Artemisa: An eco-driving assistant for Android Os. In *IEEE International Conference on Consumer Electronics - Berlin (ICCE-Berlin), 2011*, pages 211–215, 2011.

[47] R. Araujo, A. Igreja, R. de Castro, and R.E. Araujo. Driving coach: A smartphone application to evaluate driving e cient patterns. In *2012 IEEE on Intelligent Vehicles Symposium (IV)*, pages 1005–1010, 2012.

[48] Y.L. Murphey, R. Milton, and L. Kiliaris. Driver's style classification using jerk analysis. In *IEEE Workshop on Computational Intelligence in Vehicles and Vehicular Systems, 2009. CIVVS '09*, pages 23–28, 2009.

[49] Truong, Alice. "A New Take on Auto Insurance by the Mile". *Fast Company*.

[50] One-of-a-Kind Car Insurance Program Lets Drivers Save Big Bucks Based on How They Drive

[51] http://www.insurancejournal.com/news/east/2012/10/23/267659.htm In surance Journal: Allstate's Usage-Based Auto Insurance Expands to New York, New Jersey

[52] J. Paefgen, T. Staake & F. Thiesse, "Resolving the Misalignment between Consumer Privacy Concerns and Ubiquitous IS Design: The Case of Usage-based Insurance", International Conference on Information Systems (ICIS), 2012

[53] P. Handel, I. Skog, J. Wahlstrom, F. Bonawide, R. Welsh, J. Ohlsson, and M. Ohlsson: Insurance telematics: opportunities and challenges with the smartphone solution, Intelligent Transportation Systems Magazine, IEEE, vol.6, no.4, pp. 57-70, winter 2014, doi: 10.1109/MITS.2014.2343262

[54] Iqbal & Lim, "A Privacy Preserving GPS-based Pay-as-You-Drive Insurance Scheme", International Global Navigation Systems Society, 2006