A MOBILE SENSING FRAMEWORK FOR AUDIENCE EMOTION ANALYSIS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

ELDJON KEPUCKA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

SEPTEMBER 2014

A MOBILE SENSING FRAMEWORK FOR AUDIENCE EMOTION ANALYSIS

Submitted by **Eldjon Kepucka** in partial fulfillment of the requirements for the degree of **Master of Science in Information Systems, Middle East Technical University** by,

Prof. Dr. Nazife Baykal ____________________
Director, **Informatics Institute**

Prof. Dr. Yasemin Yardımcı Çetin ____________________
Head of Department, **Information Systems**

Assoc.Prof. Dr. Alptekin Temizel ____________________
Supervisor, **Work Based Learning Studies, METU**

Examining Committee Members:

Assoc.Prof. Dr. Altan Koçyiğit ____________________
Information Systems, METU

Assoc.Prof. Dr. Alptekin Temizel ____________________
Work Based Learning, METU

Assist.Prof. Dr. Cengiz Acartürk ____________________
Cognitive Science, METU

Assist.Prof. Dr. Erhan Eren ____________________
Information Systems, METU

Assoc.Prof. Dr. Banu Günel ____________________
Information Systems, METU

Date: 16.09.2014

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Eldjon Kepucka

Signature : __________________

ABSTRACT

A MOBILE SENSING FRAMEWORK FOR AUDIENCE EMOTION ANALYSIS

KEPUCKA, Eldjon

Master, Department of Information Systems

Supervisor: Assoc. Prof. Dr. Alptekin Temizel

September 2014, 90 pages

The main objective of this thesis is to develop a multi-modal framework which facilitates simple data collection using mobile sensing on smartphones from an audience, for the duration of an experimental study. Current solutions primarily rely on custom mobile sensing platforms which are expensive to develop and complicated to apply. While there are a number of mobile sensing platforms developed for smartphones targeting different domains, such as transportation and air pollution they are not designed to be used for simultaneous and synchronized data collection. The mobile sensing framework introduced in this study is focused primarily on addressing efficiency and reliability issues considering the limited resources of smartphone devices. The applicability of the framework spans cross research domains such as: emotion analysis, activity sensing, human body monitoring, and user-computer interaction. Moreover, as demonstrated in our pilot study, the introduced framework can find practical usage in industries such as advertisement and content evaluation. The applied design solution was based on the classical server-client architecture paradigm and the concrete implementation of the framework expresses high performance efficiency and comprehensive reach affirmed by our performance tests. The pilot study organized during FIFA World Cup 2014, where an audience of people was invited to watch two football matches, demonstrated the validity of our framework for real-life studies. Preliminary analysis exposes the potential of the acquired sensor data in targeted domains.

Keywords: Mobile Sensing, Emotion Recognition, Audience Analysis, Framework Design.

ÖZ

İZLEYİCİ ALGILARININ ANALİZİ İÇİN BİR MOBİL ALGILAMA SİSTEMİ ÇERÇEVESİ

KEPUCKA, Eldjon
Yüksek Lisans, Bilişim Sistemleri Bölümü
Tez Yöneticisi: Doç. Dr. Alptekin Temizel

Eylül 2014, 90 sayfa

Bu tezin amacı, deneysel çalışmalarda akıllı telefonlar yardımıyla mobil algılama kullanılarak izleyicilerden deney süresince veri toplayabilen çok-kipli bir çerçeve geliştirmektir. Günümüzdeki halihazırdaki çözümler geliştirilmesi pahalı ve uygulaması karmaşık olan mobil algılama platformlarına dayanmaktadır. Akıllı telefonlar için ulaştırma ve hava kirliliği gibi faklı alanları hedefleyen mobil algılama platformları geliştirilmiş olsa da bunlar eş zamanlı ve senkronize veri toplamak için tasarlanmamıştır. Mobil algılama tabanlı bir çerçevenin geliştirildiği bu çalışma mobil cihazların kısıtlı kaynaklarını gözönünde bulundurarak, verimlilik ve güvenilirlik problemlerine odaklanmaktadır. Geliştirilen sistem; duygu analizi, etkinlik algılama, insan vücudu izleme ve insan-bilgisayar etkileşimi gibi farklı araştırma alanlarında uygulanabilirliğe sahiptir. Bu alanların yanısıra, yapılan pilot çalışmadan da anlaşılacağı üzere geliştirilen sistem reklam ve içerik değerlendirilmesi gibi endüstriyel problemler için de kullanılabilir. Gelişitirilen sistemin mimarisi klasik sunucu-istemci tasarım paradigmasına dayanmaktadır. Somut uygulaması yapılan bu sistemin yüksek performans verimliliği ve kapsamlı bir erişime vurgu yaptığı performans test sonuçlarımızda gösterilmiştir. Tasarlanan sistem, fiziksel durumları ölçebilen mobil sensörlerin tümünden veri toplamayı sağlaması nedeniyle çok-kiplidir. Performans testleri sistemin güvenirliliğini ve genel kullanım için potansiyelini ispatlamaktadır. 2014 FIFA Dünya Kupası sırasında futbol maçı izlemeye davet edilen izleyiciler ile yapılan deneme çalışmaları, geliştirilen sistemin gerçek hayat problemleri için uygulanabilir olduğunu göstermiştir. Yapılan ön analizler, elde edilen verilerin sistemin çalışmasının amaçlandığı alanlarda kullanılma potansiyelini ortaya koymuştur.

Anahtar Kelimeler : Mobil Algilama, Algi Tanimlanmasi, Kitle Analizi, Sistem Tasarimi

ACKNOLIDGEMENTS

I want to express my gratitude to everyone that has supported me for the successful completion of this thesis. The deepest appreciations go to my supervisor Assoc.Prof. Dr. Alptekin Temizel, whose guidance and encouragement has been a key factor not only in the context of this study but most importantly in the broader perspective of shaping my academic background and providing the critical perspective so relevant for a scholar.

I am very grateful to Ferhat Kutlu who provided the important insights for addressing core requirements of our framework, and persistently and tirelessly assisting us in organizing the pilot study even during late hours of summer evenings.

Finally, I am very grateful to my family. They never let me down.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOLIDGEMENTS ix

TABLE OF CONTENTS x

LIST OF TABLES xiii

LIST OF FIGURES xiv

CHAPTER

1 INTRODUCTION 1

1.1 Motivation 1

1.2 Scope 3

1.3 Significance 5

1.4 Outline 5

2 INTRODUCTION TO EMOTION RECOGNITION 7

2.1 Emotion Recognition 7

2.1.1 Facial Expression 7

2.1.2 Body Gestures 8

2.1.3 Speech Analysis 9

2.1.4 Biological Indicators 10

2.1.5 Multi-Modal Systems 10

2.2 Crowd Sensing 11

3 REVIEW OF MOBILE SENSING FRAMEWORKS 13

3.1 Introduction to Mobile Sensing 13

3.2 Mobile sensing applicability 15

3.3 Modern Smartphones' Sensor Set Configuration 16

3.4 Mobile Sensing Frameworks and Platforms 18

3.4.1 FUNF 19

3.4.2 EmotionSense 19

3.4.3 BeTelGeuse 20

3.4.4 AnonySense 20
4 FRAMEWORK DESIGN 23
4.1 Framework’s Major Objectives 24
4.2 Design Principles 29
4.3 Framework’s Server-Client Design 29
4.3.1 Server Component 32
4.3.2 Mobile Client Component 45
5 FRAMEWORK IMPLEMENTATION 49
5.1 Implementation Analysis 49
5.1.1 Server Side Implementation Decisions 49
5.1.2 Client Side Implementation Decisions 50
5.2 Server Side Implementation Details 51
5.2.1 Database Server 52
5.2.2 Sensor Data Directory System 53
5.2.3 DB Persistence 53
5.2.4 Server’s Core Components 54
5.2.5 Web Based Client Module 58
5.3 Application Server 60
5.4 Client Side Implementation Details 61
5.4.1 Core Objects 62
5.4.2 Communication Handler 63
5.4.3 Sensor Handler 64
5.4.4 Server Communication Components 64
5.4.5 GUI Module 64
6 SURVEYS AND PERFORMANCE TEST ANALYSIS 67
6.1 Performance Testing 68
6.1.1 Device Categorization 69
6.1.2 Application Test Setup 70
6.1.3 Application Test Evaluation 71
6.2 Surveys Evaluation 74
6.2.1 Survey Setup 74

6.2.2 Overall Framework Performance....76
6.2.3 Data Evaluation....78
7 FUTURE WORK AND CONCLUSIONS....83
7.1 Conclusions....83
7.2 Contributions....85
7.3 Future Work....85
APPENDIX....87
BIBLIOGRAPHY....91

LIST OF TABLES

Table 1: The comparison of the selected database servers. 52
Table 2: Application Servers Comparison. 61
Table 3: Smartphone Devices Performance Categorization. 69
Table 4: Test Configurations for Framework Performance Evaluation. 71
Table 5: Results-Test No.1 72
Table 6: Results-Test No.2 73
Table 7: Results-Test No.3 73

LIST OF FIGURES

Figure 1: Mobile Sensing Applicability [6]. 3
Figure 2: Device physical axes definition. 18
Figure 3: System's Server-Client Design 31
Figure 4: Relational Database Module Design 33
Figure 5: Data Directory Structure for Sensor Records 36
Figure 6: Framework Component Design Overview 40
Figure 7: Server-Client Socket Protocol 43
Figure 8: Client Component Design 47
Figure 9: Database Object-Mapping Model 53
Figure 10: Design Details of Socket Handler, Soap Web-Services, Sensor Data Manager, and Request Handler modules. 54
Figure 11: Request Handler's Class Diagram Design. 55
Figure 12: Socket Handler Module's Class Diagram Design. 56
Figure 13: Sensor Data Manage Module's Class Diagram. 57
Figure 14: Web-Service Module's Class Diagram. 58
Figure 15: Web Client Module's Class Diagram. 59
Figure 16: A Base User-Interface Page. 60
Figure 17: Android Application Class Diagram. 62
Figure 18: (a) Communication Handler Class Diagram (b) Core Objects. 63
Figure 19: (a) Sensor Handler and (b)Server Communication Class Diagrams. 65
Figure 20: GUI Module Class Diagram. 66
Figure 21: Environment Setup of Emotion Recognition Experiments. 77
Figure 22: Experiment 1 – Linear Accelerometer Graph on Y-axis for two participating devices between 73rd and 83rd minute. 78
Figure 23: Experiment 1 – Accelerometer Graph for all axes of a single device between 36th and 42nd minute. 79

Figure 24: Experiment 2 – Rotation Sensor Graph on all axes for a single device between 30th and 34th minute. 80
Figure 25: Experiment 1, Accelerometer data for 2 devices on a specific goal timeframe. 81
Figure 26: Experiment 1, one single device's sensor data on: (a) accelerometer, (b) sound, (c) linear accelerometer, (d) magnetometer, (e) orientation, and (f) rotation vector.. 88
Figure 27: Experiment 1, Accelerometer data for 5 devices: (a), (b), (c) and (d)...... 89
Figure 28: Experiment 2, Accelerometer data between 34th and 60th minute for 7 devices: (a), (b), (c), (d), (e), (f), and (g).. 90

ABBREVIATIONS

OS – Operating System

IDE – Integrated Development Framework

GUI – Graphical User Interface

SD Card – Secure Digital Card

TCP – Transmission Control Protocol

UDP – User Datagram Protocol

CSV - Comma Separated Values

IMEI – International Mobile Station Equipment Identity

MD5 – Message-Digest Algorithm

SSL – Secure Socket Layer

VM – Virtual Machine

API – Application Programming Interface

HTTP – Hypertext Transfer Protocol

WWW – World Wide Web

SOAP – Simple Object Access Protocol

SQL – Structured Querying Language

OO – Object Oriented

OOP – Object Oriented Paradigm

RAM – Random Access Memory

JSON – JavaScript Object Annotation

JSF – Java Server Faces

MVC – Model View Controller

EAO – Entity Access Object

CPU – Central Processing Unit

GPU – Graphics Processing Unit	
ANR – Application Not Responding	
GC – Garbage Collector	
MAC – Media Access Control	

CHAPTER 1

INTRODUCTION

1.1 Motivation

During last decade we have been witnessing a significant switch from classical desktop and laptop computing to ubiquitous computing represented today by smartphone devices. This major shift has had a compelling impact on various industries and research domains specifically with the introduction of the physical state measuring sensors and modern mobile operating systems which collaboratively have constituted the paradigm of *mobile sensing*.

Mobile sensing refers to the capability of current smartphone devices to capture physical state data by a set of mobile sensors. The incoming information based on these sensors such as location, speed, acceleration, direction, tilting, and positioning has proven to be useful in different studies in domains such as: policy making [1], healthcare [2] [3], transportation [4], and social networks [5]. A larger set of applicable domains is introduced in [6] and visualized in Figure 1. Taking advantage of the current widespread usage of smartphones, these systems have commonly demonstrated the potential to reduce the costs related to hardware development, experiment setup, and data acquisition process. However, the applicability of mobile sensing may span further into domains such as: emotion analysis, activity sensing, body tracking and user-computer interaction. Current state of these domains provides limited research studies regarding the introduction of mobile sensing as an alternative data source. For instance, among numerous scientific methodologies, elaborated in the emotion recognition domain, based on various cues: facial expressions [7] [8], body movements [9] [10], speech signals [11] [12] [13], and biological indicators [14] [15] [16], only studies such as [17] have introduced the capabilities of mobile sensing in this particular domain. A greater impact can be seen in activity sensing and human body tracking as introduced in the following studies: [18] [19] [20].

Overall, these studies share a common characteristic: the experimentation is realized on each individual separately and the community based approach, which within the scope of this thesis we refer to as *audience,* is marginally explored. The main reason of this disproportion is related with the high cost of the implementation of systems capable of capturing data from an audience of people specifically if the architecture of these systems requires custom hardware. Recently, a steady shift is observed towards utilization of smartphone devices as a replacement to custom hardware. However the focus has still been on the *individual* as the object of the study. Therefore, our primary goal is to provide a tool for *collective* gathering of sensor data from an audience of people simultaneously. This approach introduces great advantages when targeting the application of the research on real life experiments, as we intend to achieve in the context of this thesis.

This study aims to demonstrate the positive impact of mobile sensing in multiple domains where the target of the experiment consists of an audience of people. The final output incorporates a multi-purpose framework (i.e applicable to different experiment types) capable of capturing multi-sensor smartphone data in a synchronized fashion from an audience of people, controlled from a single remote location during real life events. The framework is capable of capturing multi-sensor data in a synchronized fashion allowing harmonized data acquisition from all the participants of an experiment. Furthermore we highlight the impact of mobile sensing in the general context of effort reduction regarding system development, and simplification of the experiment organization process. It is critical to introduce a framework which can be applied to real life events away from the strict experimental environment that the majority of current frameworks are based on. This is demonstrated in our pilot study in the context of which we provide significant insights regarding the applicability of sensor collected data in the emotion analysis domain as our main focus during experimental evaluation of the framework.

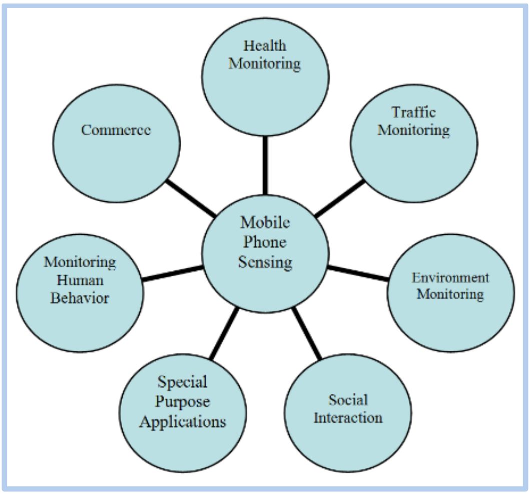


Figure 1: Mobile Sensing Applicability **[6]**.

1.2 Scope

The main objective of this thesis is to introduce an alternative mobile sensing framework targeting simultaneous data collection from multiple-sensors in a synchronized fashion from an audience of people, as opposed to conventional per person basis approach. Furthermore, the framework ought to express high reliability and efficiency in order to minimize the effect on the overall performance of the smartphone devices and enable collecting data through the whole duration of an experiment typically lasting a few hours. This definition provides the necessary insights regarding framework's scope, capabilities, and restrictions.

The framework is intended to unlock the mobile sensing capability of participants' smartphones which significantly reduces the development costs of similar systems which are based on custom made hardware and platforms. In the context of a single device, the ability to collect data from all available sensors particularly from physical state parameters measuring ones is crucial. On the other hand, use of off-the-shelf smartphones imposes a number of restrictions narrowing down the pool of potential participants to individuals possessing smartphone devices and limiting the data acquisition sources as only the sensors available on modern smartphones could be used. The sensors that we are interested in belong to the physical state measuring category and other hardware such as: light level and proximity sensors which do not provide any relevant data regarding physical state of the device. Additionally we have focused on the microphone as the sound level measuring device which provides pertinent data particularly for the emotion analysis domain and human-computer interaction. For simplicity, in the context of this study, we have defined the microphone as well in the category of physical state measuring sensors. Although smartphones are equipped with camera devices, the video modality is excluded from the data collection capabilities of our framework. Video modality was found not feasible as it introduces serious implications regarding data acquisition and static smartphone positioning toward user's facial direction.

Another major goal is to develop a multi-purpose framework that can be used in a large number of domains as descripted in the previous section. Primarily we target data collection during social activities restricted to a duration of few hours and involving a group of people. The type and scope of such activities may include: (1) spectator based activities such as: sport tournaments, theater and cinema shows, musical events where individuals share the same location, (2) individual based activities such as body movements tracking during: physical exercising and bicycle riding, and (3) gesture movements during live actions with computer devices. We mainly focus on events during which the body movements and speech are likely to be observed. Based on the activity type of the experiment, the smartphone device may be placed to a different part of the participant's body in order to capture the most relevant sensor data. For instance, during our pilot study we placed the smartphone devices on the left side pocket of participants' shirts in order to capture the movements of the whole body, particularly vertical motion and rotation.

Embedded smartphone sensors can significantly stress the battery life duration under heavy usage. Our focus during pilot testing stage primarily relies on this type of events however, the nature of the framework is multi-purpose and configurable therefore longer lasting events under different configuration characteristics, including limited number of sensors and reduced data generation rate, are feasible. This thesis hence, aims to provide the framework’s definition, design, implementation, and evaluation on real life events from the performance and data reliability perspective. We suggest the set of emotions*: joy, sadness, anger*, and *pleasure* that the acquired data might be relevant on analysis process however, the classification process lies beyond the scope of our study.

1.3 Significance

The ability to collect multiple sensor data from a group of people simultaneously in a synchronized fashion provides significant advantages for emotion analysis, activity sensing, content evaluation (i.e movie or advertisement evaluation) and human-computer interaction. This enables the analysis of data of each individual separately as well as collective analysis of the data from a number of individuals. The ability of the framework to work on participants’ smartphones and the simple use of framework are expected to have a significant impact by reducing the efforts required for experiment organization. Simplification of the experiment execution process and the use of already available smartphone devices will lead to large and more frequent data set acquisition for further data mining activities.

1.4 Outline

This thesis is organized in six chapters and each of them has the following content:

- **Chapter 2** is focused primarily on emotion recognition fundamentals and provides a literature review on major emotion recognition cues and applications. Furthermore, it highlights the notion of audience analysis as a new domain with particular interest in very recent studies.
- **Chapter 3** introduces mobile sensing and the developments that led to the wide range of smartphone’s applicability. Here major mobile sensing frameworks are identified and analyzed in details in order to identify weaknesses and address them within our proposed framework.
- **Chapter 4** provides a detailed design of our proposed framework. Specific attention is paid to the objectives of the framework and main principles that

lead the design process. Second part of the chapter focuses directly on the detailed design introducing both: server and client components.

- **Chapter 5** however, describes the implementation process specifically following the framework's design. This chapter initially introduces critical design decisions including the implementation platforms selected for each component and their benefit on overall feasibility of the proposed framework.
- **Chapter 6** defines all test results. This chapter is divided into three major parts. First part introduces performance results regarding our mobile sensing component of the framework in comparison with another state-of-the-art framework: *FUNF*. The second part introduces two real experiments applied during *FIFA World Cup 14* tournament. Furthermore, the obtained results are described and the knowledge inferred related to audience emotion analysis is highlighted.
- **Chapter 7** is the closing chapter which includes general conclusions, the final evaluation of the framework and possible future expansion areas for similar researches.

CHAPTER 2

INTRODUCTION TO EMOTION RECOGNITION

2.1 Emotion Recognition

Emotion recognition refers to the automatic classification of the emotional state of a person based on analysis of external physical expressions. As a term it has been traditionally related to sociological sciences however currently it is an interesting topic of many other fields such as computer science, image processing, voice recognition, and mobile sensing.

The role of sociological science consists of identifying a set of external human aspects which may reflect the internal emotional state and categorize them in order to identify a particular emotion. Computer science attempts to automate the entire process by building algorithms which analyze and categorize humans' expressions in order to identify particular emotions. Moreover, mobile sensing is the most recent contributor domain in emotion recognition and provides a new frontier on applying emotion recognition on an audience scale. Smartphones provide physical state measuring sensors with increasing variety and accuracies of which are progressively improved which accuracy is progressively improved. This section provide important insights regarding major cue categories that emotion recognition is based on. We describe each of them emphasizing the body gestures and speech signals which our framework intends to capture. The commonly known physical data modalities are categorized as follows: (1) facial expression, (2) body gestures, (3) speech signals, and (4) biological indicators.

2.1.1 Facial Expression

Facial expression is a core emotion indicator, therefore majority of studies in this domain exploit this emotion cue. The study introduced in [21] attempts to recognize the following emotions: anger, joy, disgust, fear, sadness, surprise and neutral state. It is one of the first fully automated facial expressions' recognition systems providing real-time face detection mechanism. Meanwhile, in [22] authors have used the same archetypes of expressions but with a specific focus on feature extraction including both detection and feature border identification. In contrast to previous researches which were based on identifying and classifying regions of interests such as eyebrows and cheeks, in [23] was introduced a different approach based on facial muscles action units defined in sociological domain in [24]. Recent techniques depend on non-conventional approaches such as the avatar based approach introduced in [25].

The setup of facial expression experiments is characterized by a static camera pointing toward the facial region of an individual and the evaluation of a single person at a time. Even though modern smartphones are generally equipped with two cameras, one on each side of the device, the injection of video data is not feasible in the context of our framework. Facial expression experiments are very sensitive toward movements and it is critical the static positioning of the camera. The scope of our framework implies data acquisition in real time events and moreover on a crowded environment and practically positioning mobile cameras statically toward the user is not applicable. The technical concerns of this aspect are its negative impact on the battery and the large size of video data acquired necessary to transmit to the remote location.

2.1.2 Body Gestures

Body gestures related studies have the following steps: identification of body movements for particular emotional states, acquisition of video or sensor data, and application of classification techniques for emotion categorization. Despite the criticism toward the accuracy of body gestures as implied in [26] [27], an increasing number of studies are observed toward this cue category specifically with the introduction of new methods of capturing data based on physical state sensors such as gyroscope and accelerometer. The authors of [28] highlight the importance of kinetics such as velocity, acceleration, and dynamics such as mass and force on emotion recognition. [29] provides insights for successfully using qualities of body movements: amplitude, speed and fluidity, for identifying basic emotions. A number of studies [30] [31] [32] [2] have already highlighted the potential of physical state measuring sensors on body gestures recognition. However, the majority of these studies are based on custom devices encapsulating multiple sensors and this is a major drawback as it introduces the necessity of manufacturing or acquisition of special purpose devices. Besides increasing the costs, this approach introduces challenges from the implementation perspective as well. Even when based on smartphones these systems are designed for different domains such as healthcare [2] or social networks [5]. They grasp data from a limited number of sensors and a single individual. Generally these systems lack the capability of integrating data from multiple sensors and do not support synchronization among multiple users. Frequently they require the constant interaction of the participants of the experiments which introduces further prerequisites prior to the experiment implementation. Despite the drawbacks, these studies provide solid fundamentals for mobile sensing applicability in body gestures identification and therefore provide substantial evidence for emotion recognition domain usage.

2.1.3 Speech Analysis

It is observed that speech contains some emotional features in the form of intonation, eloquence, and amplitude which scientists have studied in order to elaborate models for mapping emotional states to certain speech characteristics.

A large number of studies [33] [34] [35] [36] present important incentives for using speech signal for emotion recognition purposes. Again, these studies are conducted on an individual. They are not applicable to a larger audience simultaneously and moreover none of them is based on the mobile infrastructure that smartphones represent today. In our framework we include speech signal acquisition as it may provide critical information for emotion analysis and by providing multimodal data acquisition potential accuracy improvements maybe observed for emotion recognition studies.

2.1.4 Biological Indicators

Methods based on biological indicators aim to identify the emotional state of a person based on the biological data such as heart rate, blood pressure, skin temperature, and galvanic skin response. This is a relatively new domain for emotion recognition and very few studies [14] [37] [38] [39] are concentrated around internal biological indicators generally for stress level measurement. Acquisition of biological indicators related data is not feasible within the context of our framework considering the limitation of available sensors on smartphones. Even embedded temperature sensors are intended to yield only the temperature of the environment which does not provide any relevant information necessary for elaborating any emotion recognition technique. Moreover, these sorts of sensors are still expensive and not embedded within smartphone devices extensively yet.

2.1.5 Multi-Modal Systems

From the emotion recognition domain perspective, accuracy is the primary indicator of the performance of a system. Based on the results provided by studies such as [40] [41] [42] [43] multi modal systems have achieved major accuracy improvements compared to the uni-modal counterparts. However, they reflect some core drawbacks that we have repeatedly encountered in the studies throughout this chapter. They primarily serve to analyze emotional state per individual basis and do not provide any flexibility for analysis on an audience level. Moreover multi modal systems are complex and difficult to operate. They include a set of separate components such as cameras, microphones, and biosensors which are not flexible to integrate in any physical location in order to appropriately capture emotion cues of an audience during real events. Such drawbacks make it infeasible to use such systems in real life experiments as our framework targets to achieve. However, the increased accuracy by applying multi modal approach is an important feedback and our framework aims provide multi modal support. We target capturing both physical state related data from mobile sensors and speech signal from the embedded smartphone microphone.

2.2 Crowd Sensing

Crowd sensing is relatively new paradigm introduced in mobile sensing framework. At best of our knowledge it is not yet applied on emotion recognition domain and it potentially adds a new dimension on the landscape of emotion analysis. Current crowd sensing studies provide important insights specifically regarding environmental, infrastructure, and other social studies related domains.

Common Sense is one of the crowd sensing frameworks that makes use of the wide availability of smartphones to assist on analysis and solution identification of different problems related to society in general. In [44] [45] *CommonSense* is applied in combination with air quality monitoring sensors for environmental studies. However, due to its specific nature *Common Sense* does not include any other modality such as speech signal within its data capturing capabilities. Another system within the infrastructure domain is introduced in [46]. The authors show the applicability of smartphones in studies including data gathering from a crowd of people. They bring the novelty of multi-sensor data acquisition including not only geolocation but also data coming from accelerometer, gyroscope, and microphone. However, the system serves as a single mobile application and data gathering and processing is achieved within the mobile application which does not allow the flexibility of processing multiple data sets on a single remote location. Moreover, the implementation of the system is completed on *Windows Mobile 5* operating system (OS) which is relatively old and currently insignificant on the landscape of smartphones OS market.

Medusa [47] is a multi-purpose mobile sensing framework introducing an important step forward in providing a standardized way of capturing and processing multiple sensor data from smartphones. It enables the opportunistic approach within crowd sensing paradigm and enables data collection on the cloud component of the framework. However, the major issue arising with this framework is the synchronization among multiple devices. Furthermore it lacks the ability to constantly monitor the execution of tasks in mobile nodes and once the task is delivered, the central controlling node is not notified until task completion of the task by the mobile node which no room for monitoring or even adjusting the timeframe of data acquisition process. Finally the execution of crowd sensing tasks distributed by the cluster node in the cloud is not mandatory and can be executed voluntarily by the smartphone node.

Based on our survey we conclude that crowd sensing has a potential to add a new dimension to the emotion recognition domain. Crowd sensing has already been applied to a high number of domains as highlighted above and the observed results indicate the successful cost reduction and global reach when in the respective systems and frameworks, smartphones are involved. The successful integration of crowd sensing in other domains provides solid fundamentals for its usability in emotion recognition as well. However, in order to accommodate the particular requirements, such framework needs to be designed from scratch as the available frameworks lack the necessary data synchronization, and instant control over participating device, and sometimes multiple sensor data acquisition capability that the emotion recognition based on an audience requires.

CHAPTER 3

REVIEW OF MOBILE SENSING FRAMEWORKS

3.1 Introduction to Mobile Sensing

During recent decade smartphones' computing power and applicability have increased exponentially and today they have practically taken on most of the tasks previously achieved only on ordinary desktop or laptop computers. Gradually and consistently smartphones are becoming the central computing and communicating device in our everyday life [48]. Statistics show that as of the end of 2013 over 56% of people globally own a smartphone, a number significantly higher in developed countries. These statistics indicate not only the success but the unique opportunity for different industries and researchers to reach a large number of consumers instantly and obtain practically real-time data and feedbacks.

Current smartphones are complicated machines which include physical state measuring sensors: GPS, gyroscope, accelerometer, digital compass, magnetometer etc., they are enriched with network capabilities including *WiFi*, *3G* and *Bluetooth*, and run on modern mobile operating systems. In this landscape of development, researchers introduced a new paradigm of *mobile sensing*. The terminology of mobile sensing refers to the ability of a mobile device, not necessarily a smartphone, to provide raw data based on the available set of sensors. There are commonly distinguished two forms of mobile sensing: *opportunistic* and *participatory* [49] [50].

Opportunistic approach aims to automate the data collection and transmission of mobile devices with as little as possible interaction from users. Frameworks following this approach have embedded usually a complex logic on when and how any data collection would initiate, which sensors will be collecting data from, the frequency of sample collection, and the data transfer periodicity. Many aspects of the opportunistic approach require specific attention. The main risk is related to privacy infringement that might occur if there is any leak of collected data. Secondly considering the resource consuming nature of sensors' monitoring and data collecting task, opportunistic use of the device comes with the risk of the negative impact on the overall performance of the mobile device and potentially degrading the user's experience. Any framework that follows the opportunistic approach must assess the risk of privacy infringement and performance degradation at the early stage of design.

On the other side, participatory approach requires users' involvement in critical stages of the experiment. They explicitly decide at what extent they want to be involved; including decisions related to what data to be collected, of what magnitude, scope and timing intervals. Systems designed to serve the participatory mobile sensing approach require that testers have some prior technical knowledge on sensors' purpose and scope, and familiarity with the particular application.

Opportunistic approach reflects a significant superiority over participatory approach specifically on crowd sensing applications. The participation of users on application decision-making it is a drawback for both: participants and organizers of the experiment. It discourages potential participants on involving in the experiment as it requires some efforts on absorbing the necessary knowledge in order to successfully complete all tasks required. It poses a challenge for organizers on planning and conducting training sessions in large scale. Furthermore, relying on users decisions during important stages might lead to partial or complete failure of the experiment due to erroneous nature of humans. Considering these major drawbacks of participatory approach in our system we adopt opportunistic mobile sensing approach which will be described in details in the following chapter.

3.2 Mobile sensing applicability

The application and scope of mobile sensing is very wide and spreads in a large number of domains. Researchers from different domains such as: healthcare [2], social networks [5], psychology [51], and urban studies [1], are taking advantage of mobile devices to conduct experiments and acquire massive valuable datasets. Independently from the domain of study, a mobile sensing research process can be break down into three phases:

- **Hardware integration** – Consists of preparing the custom hardware, usually wearable, which would be used to provide the experiment on a certain number of individuals. Before smartphone's introduction this phase represented a challenge due to the necessity of manual assembly, programming and dissemination of the hardware. The assembly of the hardware required collecting the specific sensors and integrating them on a single board which had to have networking capabilities and local storage for data collection and delivery. Studies introduced in [32], [2] and [52] provide typical mobile sensing platforms based on customary assembled devices.
- **Data collection** – Refers to the process of registering sensors' generated data into local or remote data storages based on the hardware and software specification.
- **Data mining** – Represents the knowledge creation process based on the dataset collected by sensors involved for the experiment. Usually different domains use significantly different data mining techniques for knowledge creation.

Smartphones practically circumvent the hardware integration phase as they typically are enriched with sensing, network and data storage capabilities. Therefore researchers are saved from time consuming and expensive hardware construction perations and they can direct their resources on developing data mining techniques and find efficient means for collecting data on remote locations.

Smartphones, admittedly provide meaningful advantages for domains which involve mobile sensing as a research methodology. However, they come with some drawbacks and still cannot replace entirely custom made hardware. The first hardware limitation is the number and scope of sensors installed on a typical smartphone. Smartphones' sensors typically serve the purpose of providing physical state parameters related to device location, positioning and movements, surrounding ambient state such as noise level, ambient illumination, temperature and humidity. However, many other studies require a different variety of sensor types which are not found in todays' smartphones. The study introduced in [3] triggers the problem of required sensors which are not part of a smartphone such as air pollution measuring sensors (specifically carbon monoxide sensor). Another factor that can provide a restriction on smartphone usability to conduct an experiment, has to do with the fact that all sensors are located within a single physical device and evidentially can be placed on a single research location by restricting the researcher to track only data coming from that particular location. For human activity sensing purposes this is a significant drawback. The *BikeNet* mobile sensing system introduced in [52] or *Mercury* framework introduced in [53] make the case for the necessity of custom designed hardware where sensors are located in different part of the subject of the experiment.

3.3 Modern Smartphones' Sensor Set Configuration

Modern smartphones offer a wide set of sensors which can be categorized in sensors measuring the surroundings' state parameters and sensors measuring physical state parameters of the device. In the first category are included: microphone, camera, relative humidity sensor, temperature etc. while the second category includes: gyroscope, accelerometer, magnetic field, gravity and so on. Below we provide commonly available sensors on modern smartphones:

- **Accelerometer** – measures the acceleration force applied on three axes of the phone including the gravity force. The unit measure is m/s^2.
- **Gyroscope** – measures the rate of rotation of the device around its three axes. One single result includes three values representing the rotation rate on each axis as rad/s.
- **Gravity** – measures the gravity force applied on the device on its three axes. The unit measure is m/s^2
- **Linear acceleration** – measures the acceleration force applied on three axes of the device. In contrast to acceleration it excludes the force of gravity. The unit measure is m/s^2
- **Magnetic field** – measures the geomagnetic field on device's axes. The unit measure is μT.
- **Orientation** – measure the rotation that a device is performing around its three axes. The unit measure is $degree$.
- **Pressure** – measures the ambient air pressure. The unit measure is hPa.
- **Proximity** – measures the distance an object is located from the screen of the device. The unit measure is cm.
- **Relative humidity** – measures the ambient humidity. The unit measure is %.
- **Temperature** – measures the temperature of the environment. The unit measure is $°C$.
- **Light** – measures the ambient illumination. The unit measure is lx.

Device axes define its positional orientation in three dimensions. Each axis is defined when the device is held in its default position. The X axis is the horizontal axis pointing to the right direction, Y-axis is the vertical axis pointing up and the Z-axis points toward the outside of the screen. See Figure 2 for details.

3.4 Mobile Sensing Frameworks and Platforms

The scientific and technological communities have been proposing frameworks and implementing platforms which make it easy to capture and process sensor data from smartphones. Considering the hardware variety of smartphones and significant differences in operating systems available, it is hardly possible coming up with a single unified framework that may serve to the entire community of mobile sensing. Current devices reflect significant variations regarding the hardware and software specifications including the number and range of sensors, hardware quality, and the overall physical characteristics of the device which frequently poses a challenge if the framework includes some Graphical User Interface (GUI).

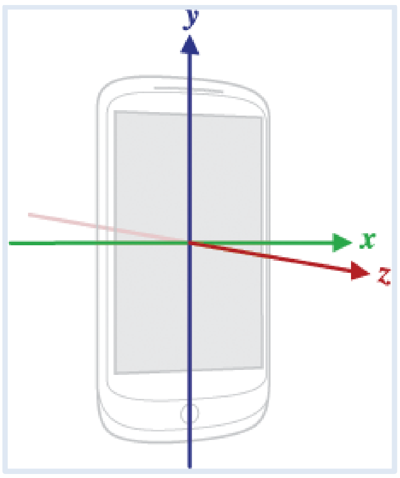


Figure 2: Device physical axes definition.

Mobile operating systems on the other side provide a heterogeneous environment which makes it practically impossible to implement cross-platform applications. *iOS, Android* and *Windows Mobile* are based on completely different operating system philosophy. They differ in adapting applications' programming languages and therefore IDEs as well. This heterogeneous environment has led researchers to provide frameworks applicable for specific use-cases with drawbacks for generic purposes. Below we will list the main and commonly accepted as the current state of the art and very successful frameworks for mobile sensing.

3.4.1 FUNF

FUNF is a modern state of the art open-source framework that provides a uniformed methodology for collecting data of any available sensor on a smartphone and processing them in a common JSON format for Android operating system. One of its core functionalities is data delivery to remote servers. It is highly configurable therefore reducing significantly the effort for application development process. During our framework analysis, development and testing we considered *FUNF* as a reference point specifically in terms of performance. As it is described in details in the following chapter, the main issue arising with *FUNF* is lack of performance for multiple sensor observation. We observed a drastic consumption of resources which were draining the battery of our testing devices in record time. Moreover, *FUNF* is focused entirely on mobile side implementation of data collection and delivery; however it does not offer any ready built-in server component for centralizing data collection. Server is a critical module for audience analysis purposes as naturally the automatic collection of data on a single location significantly reduces the efforts of the researcher and the probability for mistakes that might occur during a manual process. The initial effort for *FUNF* development is presented in [54]. Authors originally developed the framework as a system for social and behavioral sensing system based on mobile phones, nevertheless the need for similar systems in other domains lead to development of *FUNF*.

3.4.2 EmotionSense

Introduced in [51], *EmotionSense* is a platform for social psychological studies based on mobile phones. Its ability for monitoring multiple sensors concurrently is a major advantage of the system. However, it is restricted to *Symbian* operating system which currently covers less than 0.1% of the smartphones' market and it is primarily used within the research community. Considering the current mobile operating systems' landscape, the applicability of *EmotionSense* in real events is not feasible. The second major flow in functionality is lack of remote controlling of mobile devices for monitoring sensors; therefore once started the system cannot pause and in this way improve resource usage and battery life. However, the design approach of *EmotionSense* is plausible. It aims to be as autonomous as possible by requiring very little user interaction.

3.4.3 BeTelGeuse

BeTelGesuse introduces an innovative approach on building open-source modular and easy to extend platforms for mobile sensing. Introduced in [55] BeTelGeuse is characterized by modularity adopting a plug-in approach extensibility. Developed in JAVA platform it provides great flexibility on platform applicability at least theoretically as practically the current LINUX based operating systems domain is dominated by Android and applications running on pure JAVA have been frequently proven erroneous and difficult to implement. Native Android development today provides enormous advantages on many frontiers from development and access of native Android components to easiness on resolving compatibility issues arising on different available versions. Based on authors' published design and our evaluation, the highly generic nature of the platform infers a major performance drawback. In order to provide a uniform data format, the authors represent each sensor's records as String data types. By doing so, parsing operations are required. From the performance point of view such operations on *String* data types are highly expensive for both: processing unit and HEAP memory. Therefore, scaling up the platform on monitoring multiple sensors is a challenging task.

3.4.4 AnonySense

AnonySense is one of the most comprehensive frameworks introduced so far for mobile sensing purposes. Published in 2008 [56], *AnonySense* provides a multi-purpose framework for mobile sensing tasks. It encapsulates all necessary components for sensor data acquisition, local storage component, registration authority which manages eligibility of mobile devices to be assigned to certain tasks, report service for aggregating devices’ reports and uploading them to a remote server if required. Moreover, *AnonySense* addresses privacy issues arising in mobile sensing domain due to the private nature of sensor data. However, we have identified significant restrictions while exploring *AnonySense*. Firstly and most importantly the mobile phone component of the framework was written in JAVA and specifically tested on homogeneous environment of Nokia N800 devices which drastically reduces the ability of applying the application on users’ devices which are very heterogeneous. Similarly as *BeTelGeuse*, this approach fails to reduce the main cost of such experiments as requires homogeneous devices. Secondly, between the server and client components there is no continuous connection, and mobile devices periodically have to poll the task component of the framework for new tasks. Polling interval puts very visible restrictions on the usability of the framework for collecting synchronized uniform data from multiple mobile devices. By providing a minimal polling period, data synchronization error would be minimal yet the application would consume the battery very quick, meanwhile a rare polling period would thread the data accuracy by inflating the synchronization error.

CHAPTER 4

FRAMEWORK DESIGN

The scope of our study implies the design, development, and evaluation of a mobile sensing platform for audience emotion analysis during particular events such as movies, sport tournaments, and cultural shows. We focus strictly in this narrow frame of events as the emotional state of the following audience, a posteriori, is expressible and measurable based on physical state parameters. One of our main goals is to reduce the cost of such experiments, in terms of both financial and timing. Financial cost, as expressed on previous chapter, is mostly related to the hardware integration. By using smartphones we directly cut the main source of expenditures. Timing refers to the amount of man-hours for organizing of such experiments which we aim to reduce.

Both opportunistic and participatory approaches have significant flaws specifically related to the efficiency of the experiment. Participatory approach as conducted in [57] requires the user to have significant control over the experimental environment. The subject of the experiment controls the time frame of sensor data collection by explicitly indicating the beginning and the end of data collection via the available mobile application. Therefore, some initial training is required to be provided to all subjects regarding tasks' execution. In practice this might reduce the number of potential participants as this sort of experiments is primarily based on voluntary participation therefore complex and time-consuming pre-experimental procedures would lead to less people willing to participate. It is not a coincidence that mostly subjects of mobile sensing experiments are students [57] because it is difficult to attract a wider spectrum of participants based on complex and time-consuming requirements. This is critically important because focusing emotion analysis studies on a particular group of people amplifies the risk that the data acquired and the conducted analysis may belong only to the given type of experimental subjects and potentially misleading when it comes to a more heterogeneous audience which may include different age groups, professionals or even sociological profiles. Opportunistic approach on the other side takes the advantage of automaticity of data collection and removes the burden of smartphone control from the subject of the experiment, simplifying the entire pre-experimental process by lowering the participation bar. Opportunistic approach is able to circumvent the major difficulties of the participatory approach and obviously provides major advantages however they come with some drawbacks such as: (1) complexity and (2) inability to support long lasting experiments. Complexity comes as an implementation characteristic. It is a major task which requires significant amount of efforts on handling all potential scenarios which may infect the collected sensor data during application lifetime.

Our framework is based on opportunistic approach which as addressed in this chapter is significantly superior and extremely simplifies the experiment organization and management process. For the framework design we have adopted a component based approach which structures the development process and simplifies testing procedures. Our target experimental audience is restrained into single relatively short-term events typically in the range of few hours. Focusing on these particular time frames gives us the ability to listen to a large number of sensors and acquiring large amount of data which can potentially improve the accuracy of emotion sense algorithms.

4.1 Framework's Major Objectives

The specific requirements of emotion recognition domain elaborated so far in this study indicate the following objectives of our framework:

- **Access to all relevant sensors for emotion recognition available on smartphones in a reliable and robust fashion.**
  We have identified: accelerometer, linear accelerometer, gyroscope, magnetometer, orientation, rotation vector and microphone as the core

sensor group targeted by our framework which might potentially provide significant data serving the emotion recognition domain. According to this requirement our framework ought to support data collection from one or more from the above listed sensors. Moreover the sensor access frequency ought to be maximized for providing reliable data. Our framework therefore needs to provide adequate performance under the conditions of listening to all defined sensors at the maximum physical data acquisition rate.

- **Availability of the system for mid-range experiments lasting up to a few hours.**
  Addressing this requirement is particularly important and puts pressure on smartphone's resources specifically on the lifetime of the battery. Due to extensive usage of smartphone's available sensors, the battery is drained much quicker than under normal conditions; however the full battery charge should allow the experiment duration of few hours. Six is the minimum number of hours of the lifetime of a single fully charged battery under experimental conditions. This hard limit was the result of a specific analysis on types of activities which will be targeted for experimental conduction. Sport, entertainment or any other social events do not exceed the duration of few hours and the most common time span is around two hours. Overall this requirement indicates the importance of performance of our framework on possibly all targeted smartphones devices. This becomes a restricting factor if we consider the wide range of smartphones currently available. Therefore, we aim to allocate a significant amount of resources for testing our implementation approach on a large number of devices covering different performance categories.
- **Simple experiment setup and management.**
  This prerequisite emphasizes the importance of following an opportunistic approach on framework design as it requires little or no interaction from users and therefore minimum knowledge prior to the

experiment. We aim to minimize the hardware requirements and setup configuration, both steps having a direct effect on financial cost and efficiency of time management. Our intention to use participants' smartphones instead of assembling custom mobile devices significantly simplifies the experiment setup as practically researchers do not deal with any hardware configuration and/or maintenance. Nonetheless, modern frameworks on smartphones include a mobile application which needs to be distributed and installed on each device participating on the experiment. Considering that each experiment might include a different number of participants and each participant comes with a personal device, the dissemination process tends to be complicated in spite of available online distribution channels.

- **Flexibility on multistep experiment organization.**

  A core requirement of the framework is the support of experiments organized in single and multiple steps. Our literature survey indicates that available mobile sensing frameworks are primarily focused on social experiments that are conducted during a particular timeframe and cannot be expanded or applied periodically on different sample of participants or slightly different configuration of experiment's settings such as duration, location or even technical settings related to smartphone positioning, and volume calibration. Nevertheless, our framework's one of the main goals is to provide the ability of applying a single experiment repeatedly on: the same sample of participants when the effect of time is in the center of study or distinct sample of participants in order to study the differences of emotional state expression on diverse social groups. At our best knowledge, this is the first time that such requirement is put forward at the center of a mobile sensing framework. We believe that analyzing emotional state of individuals during social events is as important as analyzing the differences on emotional expressiveness of diverse social groups on similar events. Furthermore, a multistep approach towards emotion analysis experiments would bring upfront the evolution of emotional expression of a particular social group throughout relatively long periods of times (months or even years) and would make it particularly easy to study the effect of certain external experiment parameters (such as the organization time frame) on a particular sample of participants.

- **Synchronized data collection over participatory smartphones.**
  In order to provide reliable data, synchronization among participating devices is critical. Therefore we aim to introduce a synchronization mechanism that grants control over the experiment to a centralized station which ideally fits as the experiment researcher's desk. Constant control however, may be achieved by keeping alive a connection from each device to the control desk throughout the duration of an experiment. The side effect of this type of approach leads to resource usability and therefore battery duration. Moreover, constant control over participatory devices gives the opportunity to directly regulate the flow of the experiment, adjust if necessary certain parameters or even join a number of experiments or partition a single experiment into smaller fractions.
- **Centralized data collection and processing.**
  Sociological studies are very sensitive toward even minimal disruptions of acquired data. Considering the mobile nature of smartphone devices, the integrity of data is put under significant pressure. In order to minimize intentional and unintentional data disruption risks we introduce a central unit for collection and further processing. This approach guarantees the data integrity and minimized the workload of smartphone devices. The major drawback of centralized data collection consists increased complexity as communication mechanisms need to be established between the remote unit and smartphone devices. To summarize the recorded sensor data undergo the following flow:

1. **Collection** – This is the first step which necessarily needs to be conducted on each smartphone device. Data collection consists of obtaining generated sensor records and storing locally on the device either on RAM memory or available SD card temporarily.
2. **Transmission** – Our main principle on handling sensor data is to deliver in a possible immediate fashion to the centralized location. It minimizes the risk of any data infringement and assures the highest integrity of acquired sensor records from each participant. Different strategies will be considered in order to achieve data transmission by means of network capabilities of smartphones both wired (USB) and wirelessly (Wi-Fi, Bluetooth, GSM).
3. **Integration** – Acquired sensor data are very heterogeneous in both: format and statistical nature. Available smartphone's sensors generate records in accordance to their manufactures design specification. The type of the sensor logically dictates the data format as well. Typically gyroscope, accelerometer and position related sensors generate three fractional numbers respectively representing the measured value toward X, Y and Z axis. Other sensors such as light intensity or pressure provide a single fractional number. Heterogeneity of sensor data is expressed on their measured units and record generation frequency as well. Even for a specific sensor data are not necessarily received in a single chunk therefore, within a single [32]set of sensor data integration is necessary. Only after a complete union of data, it can be proceed with data analysis and result visualization.
4. **Mining** – Typically as mentioned in the previous chapters, available sensors on smartphones generate plain data which need to be further analyzed in order to infer any valuable knowledge. While it is not the main focus of this thesis, we provide a preliminary analysis of the obtained data to assess its suitability to emotion recognition domain.

4.2 Design Principles

In order to provide a comprehensive framework design we adopted few core design principles which led the design process. The framework will provide the blueprints for concrete implementations however it does not restrict the development process to any particular platform. Therefore it is crucial to provide specific independent components which have minimized correlations among each other. In this context, we adopted the principle of *separation of concerns* which introduces the necessity of a clear distribution of responsibilities among system's modules without intersections and attacks in the early stages the potential risk of fragility, rigidity, and immobility of the system. Moreover, by adopting *write only once* and *single-responsibility components* principles in the early stages of the framework's design we address the potential issues that might arise during testing and maintenance phases of the system lifecycle.

4.3 Framework's Server-Client Design

Server-Client is a very common paradigm in software engineering domain. It has become a state of the art and the default implementation approach whenever any sort of communication is required among two or more components of a system, or even among independent systems. This approach basically identifies two actors within a system based on their assigned functionalities: *server* and *client.*

- **Server** – It is the component which provides certain resources or services available for usually a large number of clients. This concept is applicable in many types of systems and does not provide any restrictions whatsoever, regarding the location of client and server components. The ordinary approach assigns clients and server to different machines belonging to different networks communicating via an established protocol however any other unconventional setup such as: both components residing of the same machine or the same network, does not violate the principles of the server-client paradigm.

- **Client** – It is the consumer component of the server-client paradigm. Based on design requirements, client sends requests to the server regarding the availability of necessary resources. Both client and server function totally independent from each other. Commonly the black-box paradigm applies on both sides: meaning that their internal procedural sequence is unknown to the other actor.

In the broad perspective, we have adopted the *server-client* approach for the high-level design of the framework. User requirements clearly state the need of a centralized method of control over devices participating on an experiment, and a common database for collecting the participants' accumulated sensor data. Therefore, the first major component of our framework will be a single server which will provide the necessary interfaces for clients to connect to, exchange the required data and disconnect. Clients on the other side will be smartphone devices therefore our client component is represented by a mobile application. A single experiment is designed to be conducted on different number of participants, which may range from tens to hundreds. Designing our framework we particularly focus on this condition as we intend to deliver a solution applicable for different types of sociological experiments.

Figure 3 provides an overview of the framework from a high perspective. Based on our design approach, server is the main component and encapsulates three abstraction layers which support two-way-communication with each other as follows:

- **Communication Interface Layer** – This is the only interface publicly available and intended to serve as a communication bridge between server and mobile clients. Communication interface offers a two way connection via *TCP/IP* protocol. The communication interface are being assigned two core responsibilities, one per each side of communication:

1. Validate every client's requests, configure and deliver them to the application layer in a reliable manner.
2. Keep track of application layer responses and distribute them to the correct clients.

- **Application Layer** – This is the heart of the framework. Application layer integrates the fundamental logic on how the communication will endure between server and mobile clients. It defines both the rules governing each connection and the level of data abstraction available to each client.
- **Data Layer** – It is the lowest level of data representation available only to the application layer. This provides the required security on the data integrity. Any direct access of mobile clients to data layer is strictly forbidden. It interchanges queries only with application layer.

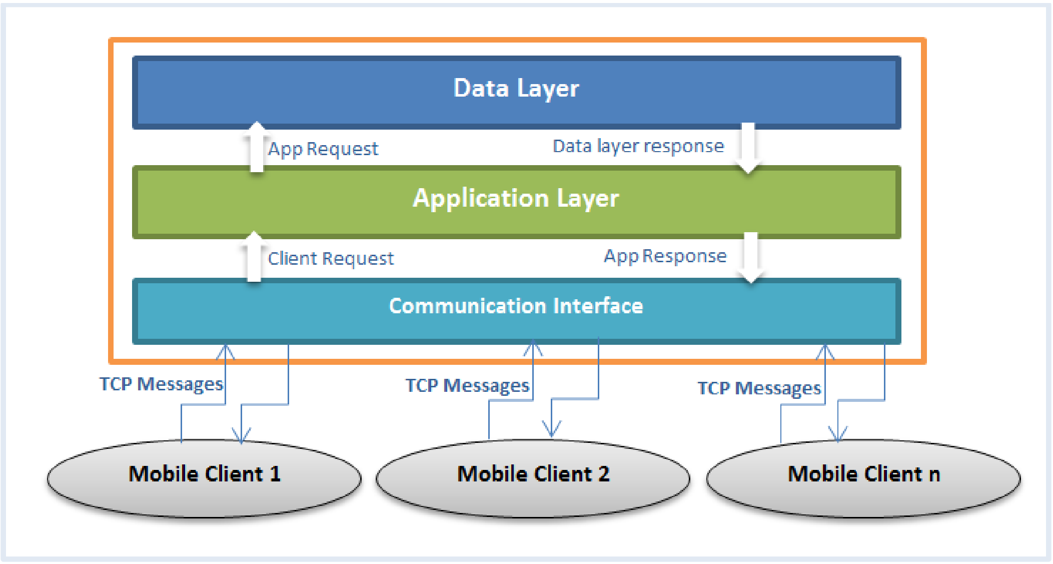


Figure 3: System's Server-Client Design

Mobile client refers to the application component of the framework running on the smartphone device and uses the *Communication Interface* to establish a connection with the server.

4.3.1 Server Component

The design of the server component is specifically a complex process due to the wide range of the assigned user requirements. Our initial observation indicates the necessity on including multiple heterogeneous modules assigned with specific tasks. Our core analysis requirement is to provide a robust framework highlighting the footsteps for the development of reliable tools assisting researches in social experiments. Our framework is focused on the narrow range of social experiments tending to analyze human emotional state during specific social events such as: sport events, musical or theatrical shows, concerts and other stage events etc. Considering the characteristics of such social experiments and the user requirements, three core modules of the server components were identified:

- **Database Module** – This is the low level manager module of any incoming or outgoing data related to the system. Considering the data types which the framework intends to handle, we have identified two database modules. The first module is the *relational database module* where any information regarding participants, smartphones and experiment characteristics will be stored. While the second module is a *directory system* where the sensor data acquired from participants' devices will be stored and accessed at any time in an understandable format by researchers.
    1. **Relational Database Component** – This module highlights the data entities such as: experiment, participant, device, and sensor, and the relations among them. We have identified four core objects relevant to the system: survey template (*survey_template* database table), instance survey (*survey_instance* database table), participant (*participant* database table) and device (*device* database table). One form of framework's flexibility is expressed in the database design introduced in Figure 4. In order to support the requirements of the ability of applying single experiment on multiple participants' samples, the survey template and survey instance notions were

introduced. A survey template defines the core attributes such as type and scope an experiment should reflect. For instance, sport tournaments and theatrical shows are being applied as different templates. A survey instance indicates a single real experiment where the sensor data is collected from a group of participants and it introduces a *many-to-one* relation with the survey template. A survey instance holds important information specifically related to the time settings (interval duration and the beginning and ending of the experiment), location (geographical and subjective location to a point of interest), and other descriptive parameters and characteristics of the experiment.

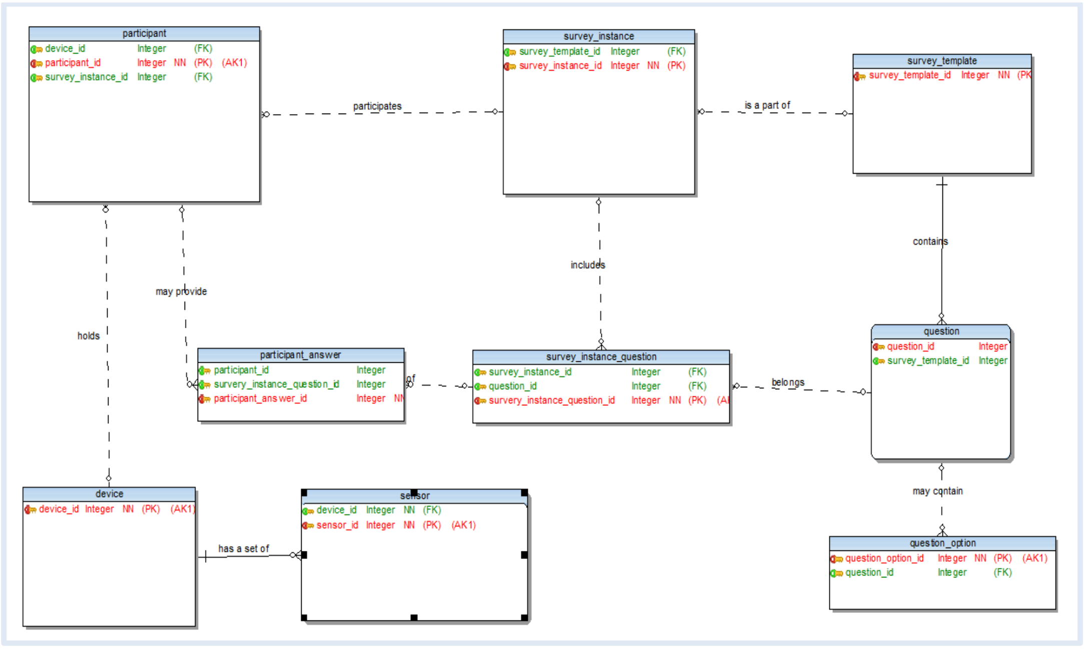

participant
device_id Integer (FK)
participant_id Integer NN (PK) (AK1)
survey_instance_id Integer (FK)
participates
survey_instance
survey_template_id Integer (FK)
survey_instance_id Integer NN (PK)
is a part of
survey_template
survey_template_id Integer NN (PK
contains
includes
may provide
holds
participant_answer
participant_id Integer
survery_instance_question_id Integer
participant_answer_id Integer NN
of
survey_instance_question
survey_instance_id Integer (FK)
question_id Integer (FK)
survery_instance_question_id Integer NN (PK) (A
belongs
question
question_id Integer
survey_template_id Integer
may contain
device
device_id Integer NN (PK) (AK1)
has a set of
sensor
device_id Integer NN (FK)
sensor_id Integer NN (PK) (AK1)
question_option
question_option_id Integer NN (PK) (AK1)
question_id Integer (FK)

Figure 4: Relational Database Module Design

Equally important is the information related to the participants of the experiment and devices applied throughout the duration of the experiment. That is the reason of the *one-to-many* relation between the survey instance and the participant database tables. In order to avoid any privacy related concerns participants of each experiment are considered as unique units and no relation can be inferred even if one participant is engaged in more than one experiment. We have established a logical *one-to-one* relation between participant and device tables of the database. Aside from these core database tables we have introduced few additional tables which play a secondary role in the overall relational database. The *sensor* table is introduced in order to store configuration setup of a device and its availability might be advantageous in order to assess the hardware differences among participating devices prior to the experiment. Meanwhile the rest of four database tables: *question, question_option, participant_answer, survey_instance_question* serve to the single purpose of voluntarily persisting information that might be relevant to the experiment from all participants. Such information is structured in a very flexible way of a simple questionnaire with questions organized in multiple choice or free response format. No other restrictions are provided whatsoever, and additionally the questionnaire is optional for both researcher and participants. It is critical to highlight the fact that no sensor data are recorded within the database. This design decision is applied due to the fact that they will not be subject to any relational querying afterward. Storing sensor data in common file formats such as *CSV* and *EXCEL* (Microsoft's Spreadsheet Application Format). The relational database entity runs on a separate module which is represented by a database server, see Figure 6. According to this design approach direct operations on data are handled completely by the database server which provides an interface for connecting to the required database and submit the necessary queries. A major advantage inferred from the introduction of a database server is the extensive modularity promoted to the framework which in practical terms means that any database server maybe inserted or replaced during implementation or even maintenance phase without affecting the rest of the system. Referring to Figure 6 the communication between the database server and the application server is significantly simplified by introducing database adapter layer which serves as an intermediary plug-in for any component to the database server.

2. **Directory System** – This component of the database module is responsible of recording all sensor data received from participants of any experiment. The main purpose of the directory system is to provide an elegant file-based structure where each file is uniquely identified and no conflicts may occur among acquired data from tens and hundreds of experiments that the system should be able to handle. In contrast to the relational database approach this practice is advantageous in terms of both simplicity and performance. Moreover, having the sensor data in common file formats simplifies further the analysis procedure as any third party application would be feasible for reading and processing these data files. The directory structure was designed in a form of multilevel tree structure resolving queries in a *O(1)* time complexity. Above the performance advantage the tree directory structure presented in Figure 5 provides most importantly the flexibility and user friendliness attributes which guarantee the total control of data by the researcher. The directory format for handling sensor data received from experiments' participants was designed to distinct experiments from each other and store separately the data of all applied experiments. The tree format stretches into four depth levels from the root directory down to the single data files. Each tree level, below the root directory refers to a lower level object from *survey_instance, device, and sensor*. The first directory level provides a list of all implemented experiments. It guarantees the uniqueness of folders' names by using a combinative naming strategy of including experiment's name and its unique id generated within the *survey_instance* database table. The following tree level of this

structure is responsible of providing separate folders per each participating device represents single files per each sensor data received during a single fragment of the experiment.

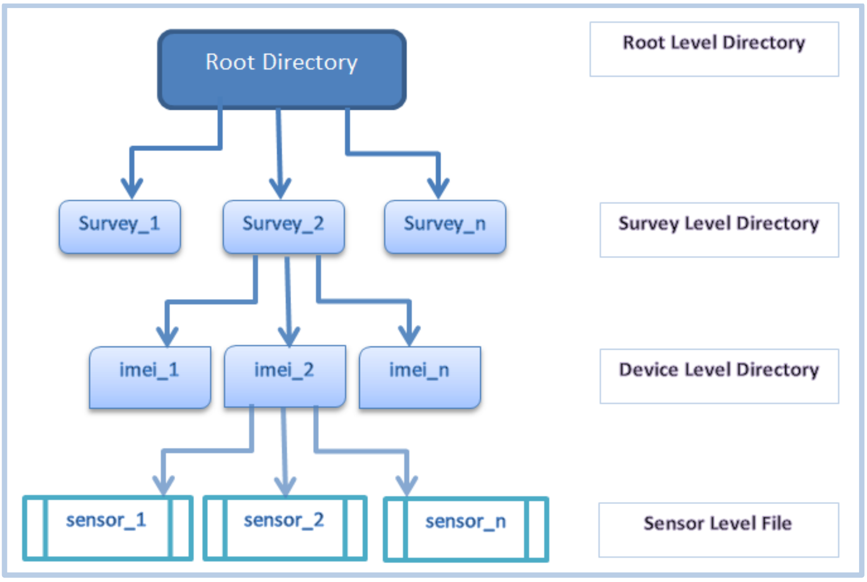


Figure 5: Data Directory Structure for Sensor Records

- **Application Server** – One critical design decision was to provide a web-based solution to the researchers for controlling the experiment’s flow, its participants and devices available for sensor data acquisition. This approach provides a high flexibility for implementation strategies due to the wide range of available open-source application servers and the extensive number of solutions that they provide in terms of application design and implementation. Modern application servers are highly scalable, optimized for delivering reliable performance, and most importantly they support multiple platforms including *Windows*, *Linux* and *MacOS* based operating systems. Most recognizable open-source application servers include: *Tomcat*, *Glassfish, JBoss*, *Jetty*, *WebSphere* and *WebLogic*. Our implementation, as it will be widely addressed in the following chapter, includes a *Glassfish* server in place of the application server component.

  The application server is part of the application layer component from Figure 3 for the server side module of our framework. It is a key integrative element of our framework as it connects the database layer with the communication interface available to client mobile applications residing on smartphone devices part of a particular experiment. Application server provides the means of reliable two-way communication between: clients and the control application, and control application and application server. At this point we may conclude that the application server presents a junction point for communication among all components of the framework. This approach of centralized communication was deliberately applied in our framework design due to its three major characteristics which amplify its superiority over a decentralized solution:

1. **Scalability** – Practically with no extra efforts and requiring minimum allocation of additional resource, supplementary devices can be part of an experiment. The infrastructure of the centralized communication provides ideal means for scaling up and down the size and scope of an experiment.
2. **Reliability** – It refers to the degree that the communication among framework's components is stable and consistent. The desired reliability is achieved by optimizing the application server and providing unified *TCP/IP* based communication means which reflect the consistency in data delivery among components. Moreover, sensor data transmissions are designed to be equipped with extra security. For this purpose we have adopted the strategy of accompanying each sensor data block with a *check-sum* datum in the MD5 format which will be delivered to the server by mobile clients attached to the sensor data. MD5 algorithm produces a 128-bit long (16-byte) hash value expressed as a 32 digit hexadecimal number.
3. **Security** – It refers to the communication security between two connection types: mobile client application with application server, and control application with application server. In case the database server resides on a separate physical machine than the communication

between database server and application server should be as secured as the previous communication types. For this purpose we have identified the SSL cryptographic protocol considering the advantage of key transmission of an asymmetric encryption. However, it is necessary to highlight that SSL provides the confidentiality for data exchange among partners however; it does not prohibit in any way third parties from listening to the established communication.

- **Web-Based Control Application** – Our requirements point toward the necessity of centralized control over the experiment on procedures such as: experiment definition, experiment initialization, sensor data collection real time governing etc. Therefore a critical design aspect of the framework is to provide an adequate solution that diverts the control over experiment's flow specifically of smartphone devices to a single control station. We considered two main approaches prior to the control application design: *web-based client* vs *desktop client*. The desktop client approach infers to the method of implementing a single application that would run in a standalone manner on a particular platform. The main advantage of this approach is related to the irrelevance of the presence of a complicated application server reducing in this way the amount of required resources in terms of memory allocated and processor cycles. Even platform independence goal is achievable by adopting *Java* related technologies on development process which characteristically run on separate layer called the VM.

  However, while evaluating the web-based approach it was clearly more beneficiary from the perspective of:

  1. **Modularity** – A desktop application solution tends to be very complex as it integrates the role of the application server with the application controller, violating the principle of modularity assigned to the framework during requirements identification stage. The web-based solution clearly distinguishes these two modules by defining fair borders on the role assigned to each module.

2. **Absolute platform independence** – Apart from the platform independence web-based approach decouples the development framework from the running platform in the contrary of the desktop approach. This fact provides significant advantages as it designates the development platform selection decision to the development stage.
3. **Guaranteed performance and security** – Adopting an application server within the framework infers some critical gains on application execution performance and most critically the maximum security during communication among multiple components. The adoption of application servers solves elegantly the security concerns regarding data transfer between client applications and the server component. Applying encryption for any communication way is highly customizable and practically every aspect is handled by the application server reducing in a compelling manner the required efforts on designing and implementing the security related solutions on a desktop application. Ultimately all major open-source application servers have been supported by major vendors, rigorously tested and iteratively improved and updated with additional features throughout many years.
4. **Wide range of development tools** – Modern application servers provide stable solutions and continuously are enriched with additional tools and today they are part of a greater ecosystem. Introducing a particular application server to the framework opens a wide horizon of platforms and technical solutions which simplify the development process. For instance, endorsing a Java-based application server opens the doors to J2EE platform (Java Enterprise Edition), GUI design API such as JSF (Java Server Faces), database mapping tools such as Hibernate etc. which will be extensively explained in the following chapter while providing a unique implementation of the framework.

5. **Improved flexibility** – The integration of application servers with IDEs and their component based approach improve the framework's flexibility especially during development phase.
6. **Global access** – Another core advantage of the web-based approach is the decoupling of the application from the hardware configuration. It does not require any installation procedure and with a properly configured application server, it is available instantly from the internet and/or intranet. From the researcher's perspective this characteristic is the most relevant one, as considering the emotion analysis experiments' nature very frequently the researcher might need to observe the ongoing study remotely.

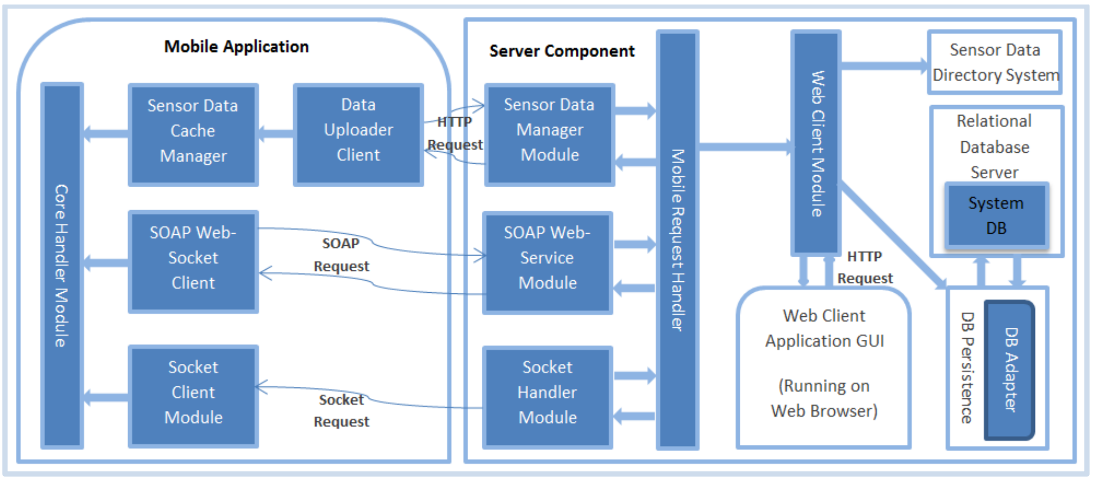


Figure 6: Framework Component Design Overview

Figure 6 provides a detailed overview of the major components of the framework. Specifically it highlights the internal design of the control application, its core modules and the communication approach established among internal components and mobile clients. The control application provides significant security on communication specifically enhanced during communication with relational database server and sensor data directory system. It has exclusive access to any database module and other components of the framework such as mobile client and web-based client can access the database module only indirectly through the control application. Residing entirely on an application server, the control application takes advantage of the availability of SSL protocol to provide encrypted communication with its external clients either mobile or web-based. The internal structure of control application is compound of the following components:

1. **Web Client Module** – This is the core module of the control application. Its major goal is to manage the requests coming from the web-client application, assess their requirements, acquire any other information from other components if necessary and construct the appropriate response for the web-client. The web client module encapsulates the business logic of the application and it is designed to communicate with the database components. Another core functionality assigned to this module resides on providing *push* notifications indicating events fired as a result of interaction established with mobile clients. Such events include: integration of a mobile client to a specific announced experiment, the beginning of sensor data collection, the end of sensor data collection, sensor data uploading status etc. It also serves as a bridge for forwarding requests coming from the web-client application. Requests of the web client application refer to commands that the researcher will provide for configuring, managing workflow, and monitoring the after-completion duration of any emotion analysis experiment.
2. **Web Client Application** – This is the application that the experiment supervisor, commonly the researcher, will operate in order to control and complete certain tasks regarding the experiment. Web client application's main purpose consists of providing a user-friendly interface to the researchers for easing the control over the experiment. Briefly the functionalities that the web client provides include: managing experiment templates and instances (inserting, editing or

removing if necessary), organizing devices participating in separate experiments, overseeing the flow of the experiment and directly controlling the timeframe of data collection and delivery from mobile clients, restrict the availability of the experiment to a desired list of smartphone devices, establish data transfer time windows, graphically notify the researcher for status of all devices during the experiment such as the connection state, data collection state, communication status with the control application and the overall capacity of devices.

3. **Mobile Request Handler** – This module is designed to assist all other modules providing communication services to mobile clients to forward their requests to a centralized location. The core purpose is to validate the clients' requests and provide an extra layer of security to the control application in order to filter any incomplete or harmful request coming from mobile clients prior to its execution.
4. **Socket Handler Module** – This and the following two components provide the communication interface feature. Their core purpose is to provide secure and reliable communication with mobile clients. Socket handler module is designed to open a socket per smartphone device for on-time governing of the experiment from the web client application. Socket commands are instantly delivered to mobile clients via *Mobile Request Handler*. Any other conventional communication approach over HTTP would be based on *polling* mechanism which consists on clients calling the server periodically. However, polling technique does not provide the ability of real-time control and frequent polling intervals will result in performance degradation and abrupt battery consumption as network actions absorb relatively a large chunk of the battery in smartphone devices. However, this solution poses a major challenge: due to its low-level programming nature it requires the establishment of a simple communication protocol between the server and mobile clients. On

their core sockets deliver only required data in terms of bytes and no additional information is included as it is the case with HTTP protocol which provides a well-structured header or other additional data included in the body of the message. The protocol designed for our framework is straightforward and is based on designing a server command consisting of the following byte sequence:

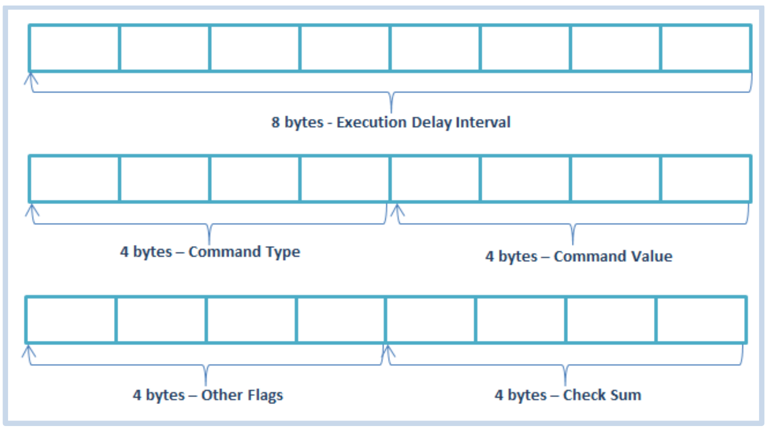


Figure 7: Server-Client Socket Protocol

The socket protocol established by the framework is simple and concise. As depicted in Figure 7 a single message of the communication socket protocol consists of 24 bytes organized in five variables: *Execution Delay Interval* – indicates the time in milliseconds that should elapse for this command to be executed, *Command Type* – indicates the category of the command with category 1 and 2 reserved for socket and data commands, *Command Value* – indicates the actual command of each category, *Other Flags* – stand for additional information that the command is associated to such as *urgent* (*0001* – binary value), *bring to top* (*0010* – binary value), *confirm* (*0100* – binary value), *repeat* (*1000* – binary value), and *Check Sum* – a security check value to be controlled by the mobile client in order to establish that the received command is complete and uncompromised.

5. **SOAP Web-Service Module** – Socket Handler Module is designed to be simple and handle commands that are ought to be instant, however it is definitely not a solution for all communication forms. Providing only socket communication between server and mobile client would extremely complicate the socket protocol introduced in the previous paragraphs and would isolate the framework from any other third party application. Therefore, we have adopted standardized communication methods via web-services for general purpose requests. Particularly we rely on *SOAP* web services in our framework for the major part of request communications with smartphone devices. Based on the security level we have defined two types of web-services: *public* and *restricted*. Statistical information such as: number of experiments, participant statistics of each experiment, metadata regarding acquired record data, experiments on pipeline, devices' specifications applied for a certain experiment, the methodology applied and the main goal stated by the experiment and so on are some of the core functionalities that the framework has defined as *public*. The restricted category is available only to smartphone devices. Each method is designed to require the IMEI of the device accessing it in order to successfully provide the required information. Common restricted functionalities include: server-client socket connection initialization, client data status update, battery state information, and so on. This approach provides great flexibility for potential information sharing strategies with other frameworks in internet.
6. **Sensor Data Manager Module** – The third step identified by the framework on data handling procedure deals with secure data transmission from smartphone devices to the centralized database location available through the server side application. Sensor data manager is organized as a distinct module and not integrated with any

of the two communication components depicted in the previous chapters. First of all they echo their personal nature which involves privacy concerns, and secondly during a single experiment it is estimated that up to hundreds of megabytes of data might be produced by a single smartphone device making the delivery process highly risky and time consuming. That is why this task will be assigned a dedicated transmission module which will not be interrupted by any other communication task of web-service or socket nature.

7. **DB Persistence** – Database persistence refers to the framework for mapping our relational database to *OO* domain model. This module simplifies the access procedures to database server. It avoids providing plain SQL queries and provides a more *OO* approach which follows the overall *OOP* of our framework. This module introduces flexibility in handling database operations and most importantly reduces development time and maintenance efforts.

4.3.2 Mobile Client Component

Mobile client component refers to a single application installed on a smartphone device and runs throughout the experiment. Some of the core responsibilities of this component include the communication with server module of the framework, interpretation of the received commands, and most importantly sensors' data collection and delivery to the server module. Following the objectives of the mobile client component, we provided a detailed design overview is described in Figure 8, where among core components it depicts critical communication links with external actors. Client component's core modules from the bottom up perspective (from low level OS to high level communication links) include:

- **Sensor Listener** – A listener provides low level communication with the OS environment as it directly accesses the interfaces available from the OS framework. This object's main role is restricted to sensor data acquisition from the available mobile sensors.

- **Sensor Manager** – The entire set of data acquired by sensor listeners is managed by sensor manager module. This module is designed to handle formatting aspects of the acquired data and provide a common format to other modules of the system. Sensor manager is a core component which provides one way communication with *sensor cache mechanism* and handles the lifecycle of sensor listeners based on server commands acquired by *Socket Client Module* or *Soap WS Module*.
- **Sensor Cache Mechanism** – This module was introduced to handle the high volume of generated sensor data. The main purpose of this module is to balance the IO operations' performance. The cache size is designed to be established dynamically based on the platform configuration. It serves as an intermediary storage facility prior to the permanent file storage of sensor data.
- **Data Storage Handler** – This component manages all interactions with the *Local Storage Directory* therefore it is the only module that has direct access to the acquired data. It provides critical functionalities for storing the data received by the *Sensor Cache Mechanism* in the appropriate data files, guarantees the data integrity and provides safe access for the *Data Uploader* components at the data transmission time frame.
- **Data Uploader** – The main role of this module is to handle the secure sensor data delivery from smartphone devices to the server component.
- **Soap WS Module** – This module stands for the client side implementation of SOAP web-services provided by the server component
- **Socket Client Module** – Naturally it represents the client side end of the socket connection established with the server side. Core responsibilities of this module include the socket connection establishment, server command acquisition and proper interpretation based on the framework's protocol introduced in the previous sections, and command deliver if necessary to the

Sensor Manager module indicating certain action such: as data acquisition initiation or complete that the server has required.

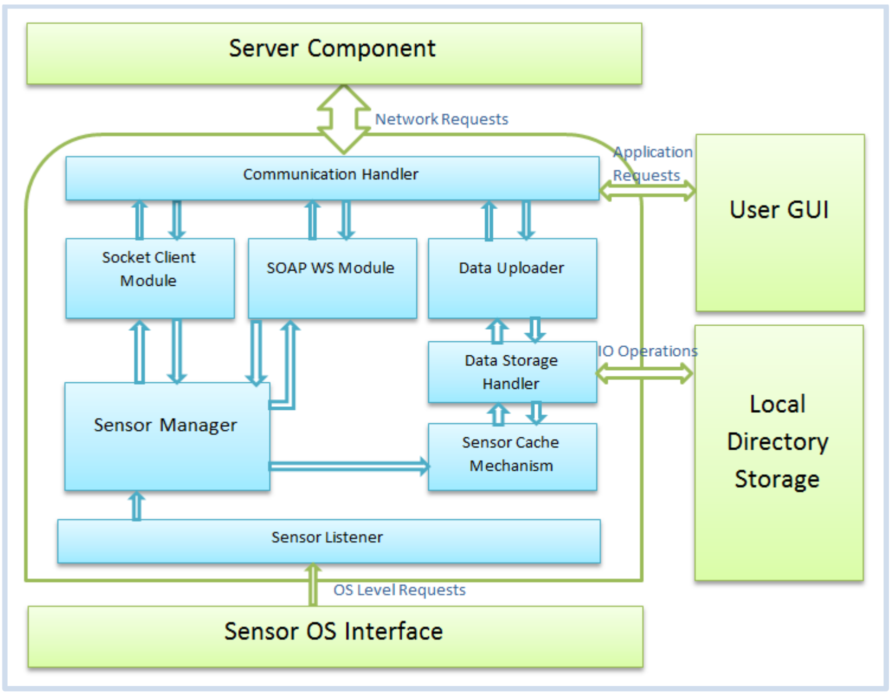


Figure 8: Client Component Design

- **Communication Handler** – In order to organize and protect the different components responsible for client-server communication, the *Communication Handler* was introduced. This single module manages the network traffic to and from the server component. It plays a critical role on securing the communication layer and validating client and server packages before processing it further in the application hierarchy.

Practically, as both: server and client components demonstrate, the network communication is a core activity handled by many modules. This fact combined with the private nature of generated sensor data, poses a challenge from the data privacy perspective which we address early in the design phase by introducing encryption in data communication. All client to server communications are done via the SSL protocol which guarantees the privacy and integrity of the data. Moreover, any collected information that serves as a direct or indirect identifier of the participating smartphones (IMEI number and MAC address) is hashed before stored on the server’s database by MD5 algorithm. This procedure guarantees the anonymity of the participants and decouples the stored sensor data from participating smartphone devices.

CHAPTER 5

FRAMEWORK IMPLEMENTATION

The framework's design proposed in the previous chapter underlines the major guiding principles and the core attributes that a framework based on mobile sensing for audience emotion analysis should consist of. This chapter provides a detailed description of the entire implementation process focusing on the adopted technologies, implementation details, performance attributes, and privacy issues. The opening sections focus on the general technological approach and the rationale behind some core implementation decisions. It continues with a detailed implementation of core modules according to the framework's design.

5.1 Implementation Analysis

For server component's implementation we have considered two major platforms: *Java* and *.Net*. Meanwhile the smartphone domain is slightly less homogeneous and more vendors provide unique platforms specifically regarding the operating systems: *Android, iOS, Windows Mobile (Windows 8), Blackberry, Symbian* etc. The framework design does not introduce any restrictions on the development tools whatsoever. From the academic perspective we have adopted two main development principles: (1) the development process has to rely exclusively on open source tools and frameworks and (2) our implementation output ought to be inclusive towards possibly all platform and mobile devices. Considering our core principles we analyzed each main component: server module and client application, separately.

5.1.1 Server Side Implementation Decisions

For the server side implementation process was decided to adopt the *Java* based platform over the *.Net* one. It provides a rich development environment (*Eclipse*

IDE) and a large open-source third party APIs. At the same time Java platform is characterized by environment independence as it runs on JVM, high performing available application servers, and it is backed by major software vendors and a large online community.

5.1.2 Client Side Implementation Decisions

Selecting the right platform for the mobile application development was a more complicated process however *Android* platform, according to our analysis, provides the best option among mobile platforms that fulfill our requirements and development principles. Currently *Android* is the most spread mobile OS among smartphones and occupies a major chunk of the smartphone market. Its major characteristics, which led us to select as the client development platform, are summarized as follows:

- **Support for any device configuration**. *Android* provides support for devices of all major vendors and any configuration including different screen size and pixel density, RAM capacity, sensor specification etc.
- **Open-Source approach**. *Android* is an open-source project available online and accessible by anyone. As such practically the entire Android source code is public and hardware vendors have access to the operating system without any financial overhead.
- **Adaptation of Java platform**. It is commonly accepted that a modern mobile operating system needs to provide a large user-based applications in order to succeed on adapting itself toward the user based community. For this purpose the Java programming language was adapted as the major development method opening the application market for a large community of Java related developers which smoothed out the Android learning curve.
- **Centralized application distribution system.** Android has drastically cut the intermediary steps from development to delivery of the application by the introduction of its community based online application distribution system: *Play Store*.

Overall success of Android platform it is translated today in a staggering amount of market share and a trend that is being consolidating year after year. Currently it is the dominating platform and different researches and market analysis [58] foresee its major role in the following stages of mobile development. If we break down the smartphone market generally *Android* is available in around 74% of the current devices, iOS in around 20%, Windows Mobile 3% and other operating systems in 3% of the available smartphone devices as till the first quarter of 2013 [59] and the trend remains advantageous for *Android*. The last statistics available report that for the second quarter of 2014 *Android* occupied 84.6% of the market share (iOS 11.9% and Windows Mobile 2.7%) [60]. Furthermore we have considered a common approach that is followed by a small number of vendors which use HTML based application development, specifically on HTML5 standard, which can be wrapped around different application formats and runnable in all major operating systems. The drawbacks with this approach however, significantly affect the performance of the application and accessing low-level device sensors it is troublesome and may fail unpredictably in any device specifically within the heterogeneous environment of Android supported devices. The introduction of HTML5 standard provided some traction for this approach to evolve yet, considering the implications in user's interaction performance major application providers have resigned toward the native implementation approach. Our mobile component heavily resides on mobile sensors which highlight the criticality of the overall application performance. This is the major drawback that excludes the adaptation of HTML based development. Ultimately, evaluating the superiority in market share, large development community, and most importantly open-source nature, we concluded that Android provides the optimized development platform for the mobile component of our framework.

5.2 Server Side Implementation Details

Following the framework's design we will provide a detailed implementation based on Java platform. This section encapsulates the implementation details for each server side component. Referring to Figure 6 our framework dictates the usability of third party component and the necessity of implementation of brand new components. The following sections provide a detailed overview for each specific server side module.

5.2.1 Database Server

One of the core modules introduced by the framework is the relational database. The design of the database from Figure 4 indicates the necessity of a well performing database server. Among open source solutions we have selected five potential candidates to take over our database server module: *Firebird*, *MySQL*, *PostgreSQL*, *Derby*, and *Drizzle*. We evaluated each of them based on: multi-platform support, update release frequency, SQL support, database size limitations, availability of supplementary tools (data management software, persistency API adapters, graphical representation tools etc.), and performance boosting techniques (B/B+ indexing trees, reverse indexing methods etc.). Each indicator is quantified in a scale from 1 (indicating poor availability) to 5 (indicating strong characteristics) and summarized in Table 1. The best performing DB according to our evaluation is *PostgreSQL* and it was selected as the database server of our framework.

Table 1: Database servers' detailed review.

	OS Support	**Release Frequency**	**SQL Support**	**DB size limit**	**Supportive tools**	**Boosting Techniques**	**Final Result**
Firebird	4	2	5	4	4	4	23
MySQL	5	4	4	4	5	4	26
PostgreSQL	**5**	**5**	**5**	**4**	**5**	**5**	**29**
Derby	4	4	5	5	4	1	23
Drizzle	2	1	5	4	2	2	16

5.2.2 Sensor Data Directory System

This is the second module that completes the database component of the framework. Considering the nature of the sensor data expected to be acquired from smartphone devices throughout experiments, the directory system address the performance issue related to the size of expected data and simplifies its availability and readability for third party applications. This module does not provide any functionality however; its maintenance is assigned to the Sensor Data Manager Module which dynamically handles the aspect of the directory system’s lifecycle.

5.2.3 DB Persistence

Our DB persistence module is supported by Hibernate API. Hibernate is a modern API which bridges the gap between database models and OO applications. For this purpose we have created an abstraction layer of Java classes representing each table of our database design. The DB persistence module consists of five major class objects introduced in Figure 9 and a single JAR component or the *adapter*. Java classes are responsible for each database table and serve as an intermediary layer for performing core database operations. The adapter serves as the intermediary communication layer with the database server which provides a particular access protocol distinct from other available database servers.

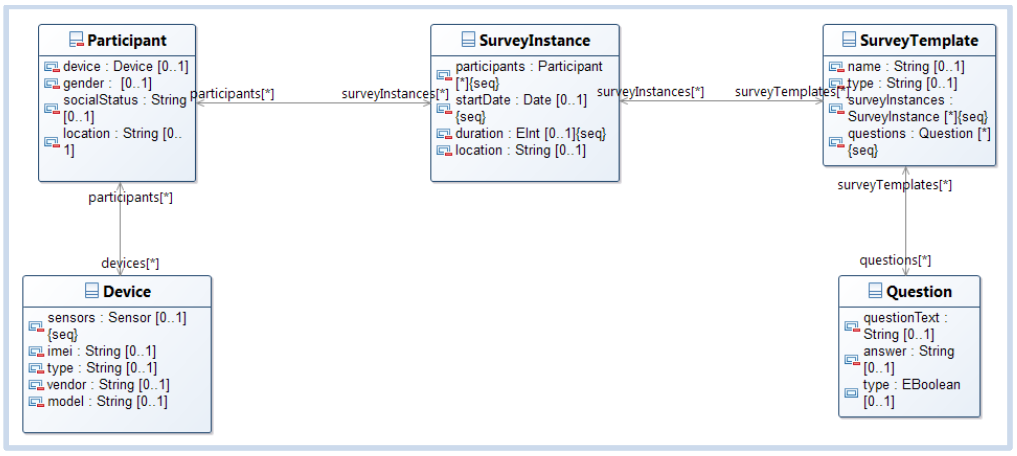


Figure 9: Database Object-Mapping Model

5.2.4 Server's Core Components

This section includes the design details regarding the four core components of the framework: *Socket Handler Module, SOAP Web-Services Module, Sensor Data Manager Module,* and *Mobile Request Handler*. A simplified but overseeing class diagram of the core modules is introduced in Figure 10.

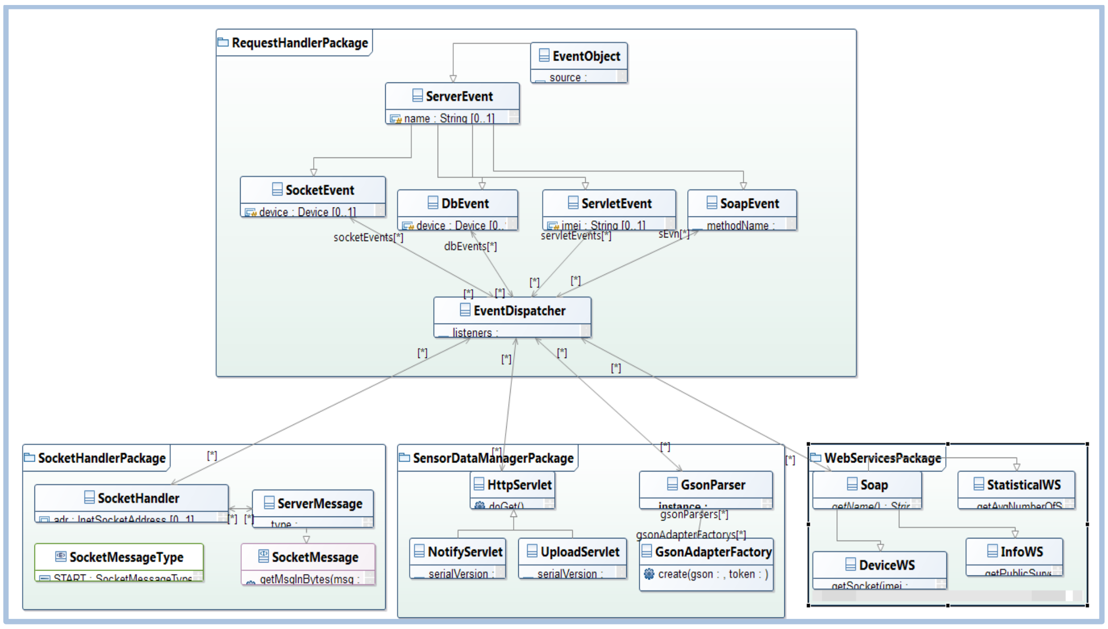


Figure 10: Design Details of Socket Handler, Soap Web-Services, Sensor Data Manager, and Request Handler modules**.**

Request Handler is responsible from directing the communication flow among other components. The basic functionality is to notify listeners of critical events generated by *Socket Handler, Sensor Data Manager*, and *Web Services* modules. Based on such events: initiation of the survey, socket connection establishment between a device and the server, web service requests' acquisition, sensor data uploads etc. the *EventDispatcher* notifies other components specifically within the *Web Client Module* in order to visualize to the researcher survey's state. Specifics on implementation design per module is provided as following:

- **Request Handler Module** – This module is assigned the task of managing the communication among other modules. As depicted in Figure 11 it encapsulates a set of event objects: *ServerEvent*, *SocketEvent*, *DbEvent*, *ServletEvent*, and *SoapEvent*. For instance, the *SocketEvent* is generated by *the SocketHandler* at key lifecycle moments including socket initiation, device connection, message delivery, message received, device disconnected, and socket closed. Web client module is notified of each event and graphical objects' state is updated accordingly.

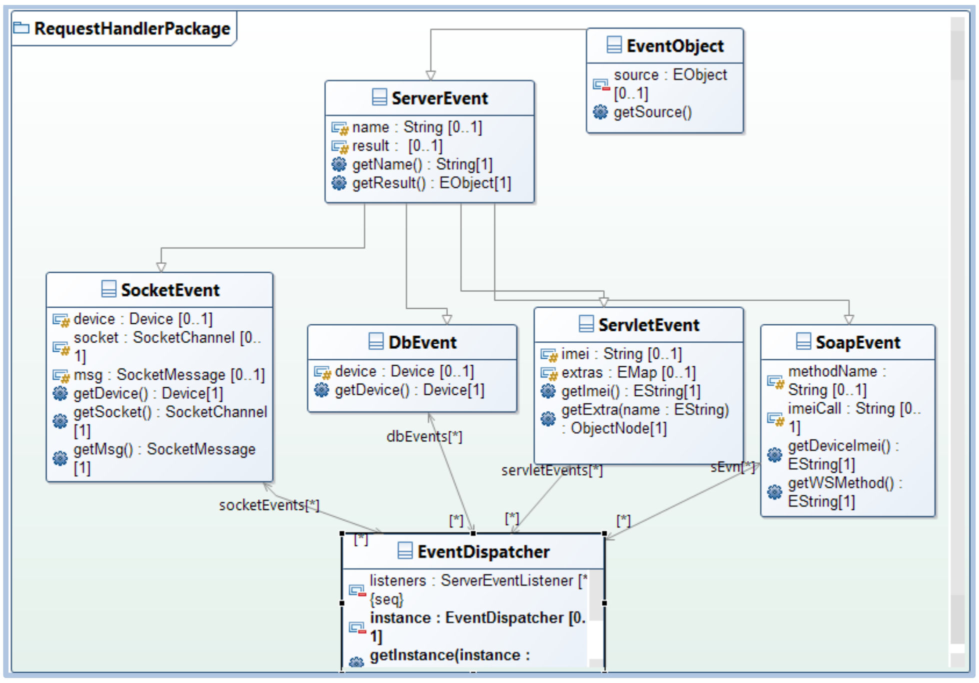


Figure 11: Request Handler's Class Diagram Design.

- **Socket Handler Module** – It manages the socket connections between the server and smartphone clients during a particular survey. In order to avoid the creation of a dedicated thread per connection we provided a solution based on *Selectors* introduced in [61]. Selectors may contract multiple connections with a single thread by implementing the *readiness selection* notion (the ability to notify processes when IO data is ready by excluding periodic polling). We introduced two more threads within the *SocketHandler* class: *WorkerThread* and *ConnThread* which respectively check received or to be sent messages to the connected smartphone devices, and the established connection with the server. Socket handler provides the instant communication necessary to control data collection from the server station. A breakdown of this module is introduced in Figure 12. *SocketHandler* is a *Singleton* class and handles the socket based connections with mobile clients. This instance delivers commands to the smartphone clients for initiating, pausing, continuing, finishing or any other command defined within the communication protocol. Typically, used by the *Web Mobile Application* (via *Request Handler* component) which serves to the researcher for directly controlling the ongoing experiment.

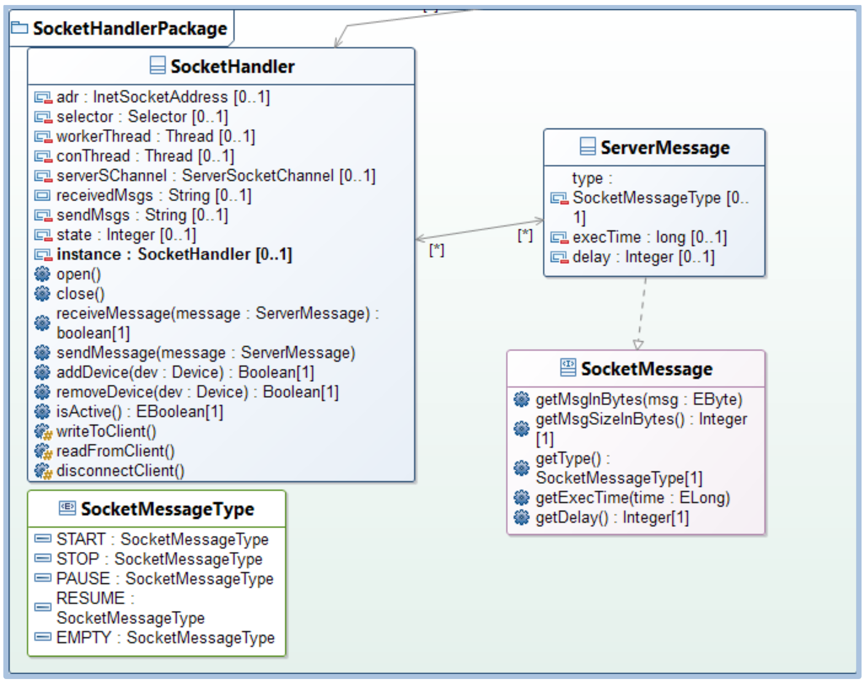

SocketHandlerPackage
SocketHandler
adr : InetSocketAddress [0..1]
selector : Selector [0..1]
workerThread : Thread [0..1]
conThread : Thread [0..1]
serverSChannel : ServerSocketChannel [0..1]
receivedMsgs : String [0..1]
sendMsgs : String [0..1]
state : Integer [0..1]
instance : SocketHandler [0..1]
open()
close()
receiveMessage(message : ServerMessage) : boolean[1]
sendMessage(message : ServerMessage)
addDevice(dev : Device) : Boolean[1]
removeDevice(dev : Device) : Boolean[1]
isActive() : EBoolean[1]
writeToClient()
readFromClient()
disconnectClient()
[*]
[*]
ServerMessage
type : SocketMessageType [0..1]
execTime : long [0..1]
delay : Integer [0..1]
SocketMessage
getMsgInBytes(msg : EByte)
getMsgSizeInBytes() : Integer [1]
getType() : SocketMessageType[1]
getExecTime(time : ELong)
getDelay() : Integer[1]
SocketMessageType
START : SocketMessageType
STOP : SocketMessageType
PAUSE : SocketMessageType
RESUME : SocketMessageType
EMPTY : SocketMessageType

Figure 12: Socket Handler Module's Class Diagram Design.

- **Sensor Data Manager Module** – This module manages the sensor data transfer process and parsing of JSON objects to Java classes which is supported by GSON API [62]. Parsing functionalities are encapsulated in two classes: *GsonParser* which provides the main parser object and *GsonAdapterFactory* providing additional procedures for non-standardized parsing, as depicted in Figure 13. They are primarily used for providing a common approach for complex data exchange via web-services benefiting from the light-weight and easy to implement characteristics of JSON format. The primary objective however, it is accomplished via *Servlet* components which are basically Java classes capable of responding to HTTP requests. Sensor data file transfer process could not be achieved by SOAP web-services alone considering their lack of support for file transfer. Even the common approach of file delivery in a BASE 64 format is not feasible for large files due to *OutOfMemoryException* risk as the available application's memory is very limited.

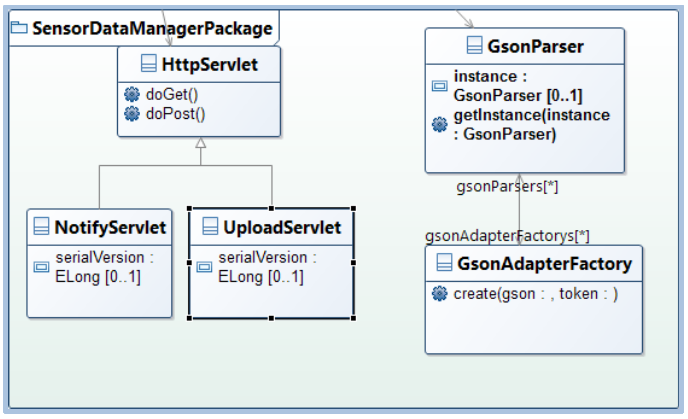
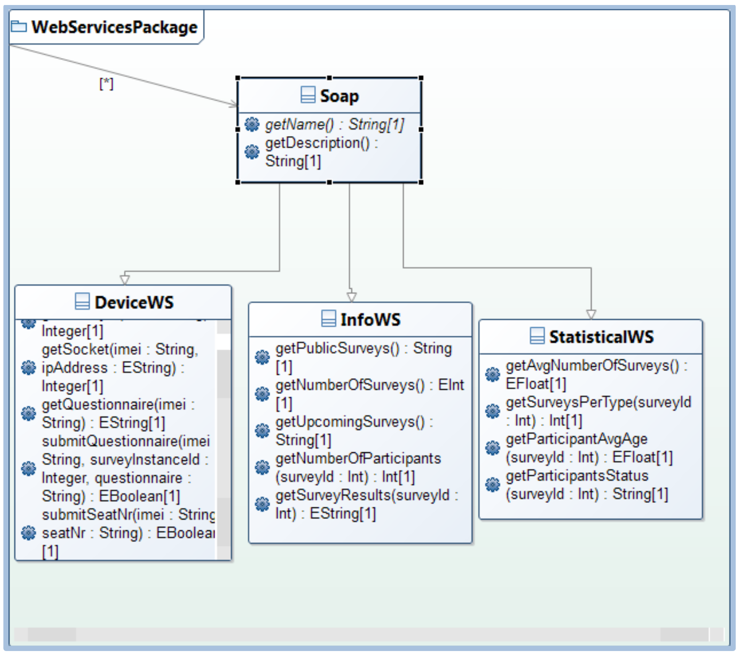

SensorDataManagerPackage
HttpServlet
doGet()
doPost()
GsonParser
instance : GsonParser [0..1]
getInstance(instance : GsonParser)
gsonParsers[*]
gsonAdapterFactorys[*]
NotifyServlet
serialVersion : ELong [0..1]
UploadServlet
serialVersion : ELong [0..1]
GsonAdapterFactory
create(gson : , token :)

Figure 13: Sensor Data Manage Module's Class Diagram.

We target file of a size of ~100 MB however, modern Android OS this limit is set to 128MB but significantly varies and a large amount of devices available on the market have a limit of 64MB and below.

SOAP Web-Service Module – There are three classes major classes that provide a set of different web service methods: *DeviceWS*, *InfoWS*, and *StatisticalWS* as described in Figure 14. Web-Services are categorized into device (*DeviceWS*), general informative (*InfoWS*) and statistical (*StatisticalWS*) group. This categorization provides a functional break down considering the wide range of web-service methods included in this module and introduces a different security level per each we-service type. *DeviceWS* provides the methods called by smartphone devices during any experiment within a secure connection. Only eligible devices (based on IMEI identifiers) are allowed to have access on: active survey, socket connection attributes, etc. *InfoWS* and *StatisticalWS* provide general information in respectively regarding public access of experiment results and overall statistical information regarding participants' categorization and number of conducted surveys, their type and scope.



WebServicesPackage
[*]
Soap
getName() : String[1]
getDescription() : String[1]
DeviceWS
Integer[1]
getSocket(imei : String, ipAddress : EString) : Integer[1]
getQuestionnaire(imei : String) : EString[1]
submitQuestionnaire(imei String, surveyInstanceId : Integer, questionnaire : String) : EBoolean[1]
submitSeatNr(imei : String seatNr : String) : EBoolean [1]
InfoWS
getPublicSurveys() : String [1]
getNumberOfSurveys() : EInt [1]
getUpcomingSurveys() : String[1]
getNumberOfParticipants (surveyId : Int) : Int[1]
getSurveyResults(surveyId : Int) : EString[1]
StatisticalWS
getAvgNumberOfSurveys() : EFloat[1]
getSurveysPerType(surveyId : Int) : Int[1]
getParticipantAvgAge (surveyId : Int) : EFloat[1]
getParticipantsStatus (surveyId : Int) : String[1]

Figure 14: Web-Service Module's Class Diagram.

5.2.5 Web Based Client Module

This module introduces a user-friendly and efficient UI application running on a web browser intended to assist researchers on controlling the course of a particular experiment. The implementation strategy of this module is based on JSF specification which introduces MVC paradigm for developing component oriented Java based web applications. Among a large number of vendors providing rich sets of JSF components *Primefaces* is highlighted as a leading API. It provides a rich set of components inside a very compact package and it requires minimal integration efforts. Within the MVC architecture, the *View* of application consists of HTML web pages available to the researcher on the web-browser. The *Model* component consists of special purposed Java classes known as *ManagedBeans* which reside on the application server and its lifecycle is managed by the *Controller* component: JSF engine. *ManagedBeans* encapsulate the business logic of the application and commonly provide database operations via *EAO* objects enclosed in a single class: *EAOManager*. As described in Figure 15, *EAOManager* provides some core generic methods for supporting all database operations per each table therefore resulting in no code redundancy and improved query execution performance.

Following the framework's design, the web client module communicates with the database and the request handler modules. Particularly *ManagedBeans* are responsible for validating and submitting queries to the database in accordance to the user's input data regarding survey instances, survey templates, devices, or participants. Respectively, submitted data are validated and processed by *StartSurveyBean*, *SurveyTemplateBean*, *DeviceBean,* and *ParticipantsBean*.

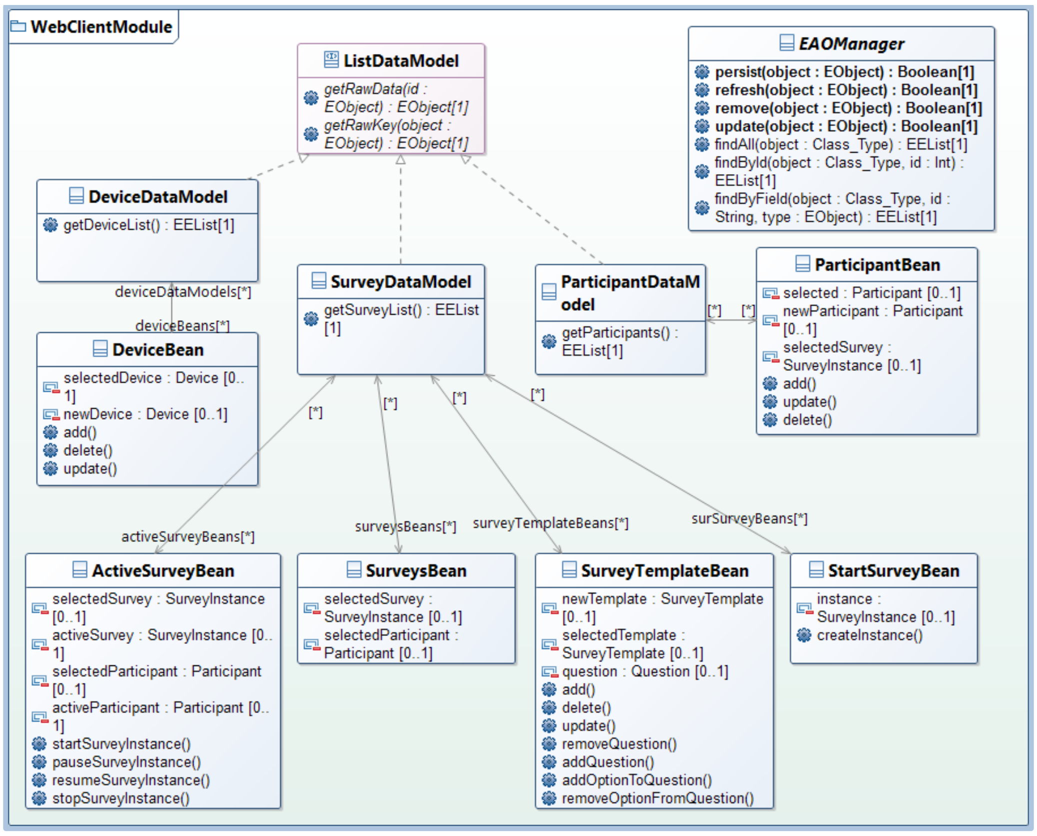
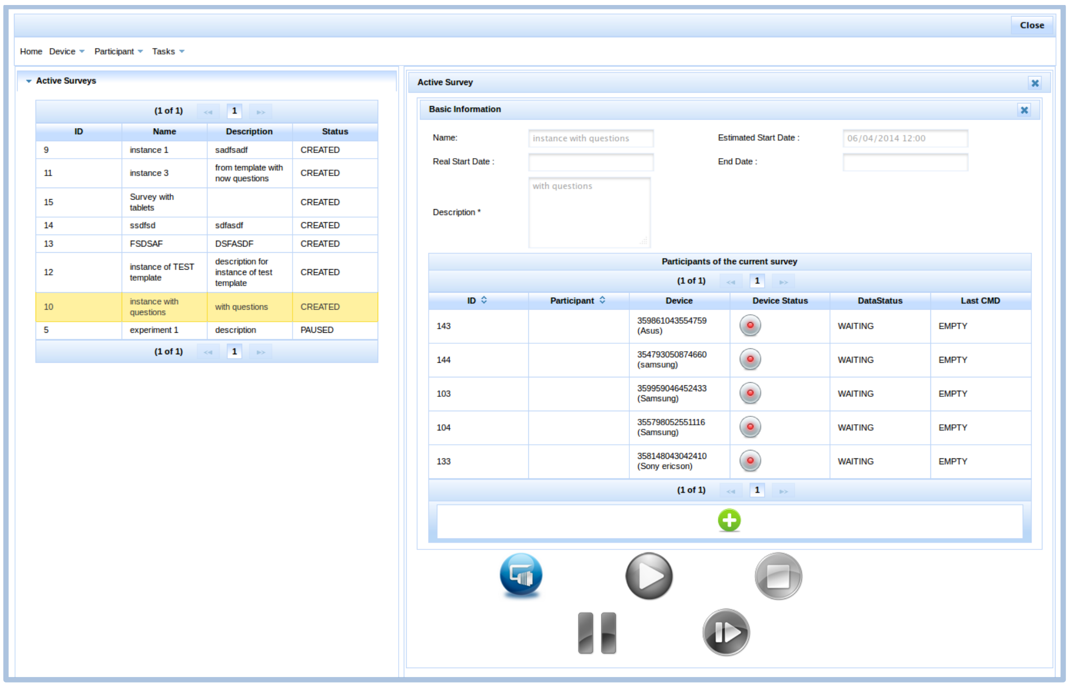

WebClientModule
ListDataModel
getRawData(id : EObject) : EObject[1]
getRawKey(object : EObject) : EObject[1]
EAOManager
persist(object : EObject) : Boolean[1]
refresh(object : EObject) : Boolean[1]
remove(object : EObject) : Boolean[1]
update(object : EObject) : Boolean[1]
findAll(object : Class_Type) : EEList[1]
findById(object : Class_Type, id : Int) : EEList[1]
findByField(object : Class_Type, id : String, type : EObject) : EEList[1]
DeviceDataModel
getDeviceList() : EEList[1]
deviceDataModels[*]
deviceBeans[*]
DeviceBean
selectedDevice : Device [0..1]
newDevice : Device [0..1]
add()
delete()
update()
SurveyDataModel
getSurveyList() : EEList [1]
ParticipantDataModel
getParticipants() : EEList[1]
[*]
[*]
ParticipantBean
selected : Participant [0..1]
newParticipant : Participant [0..1]
selectedSurvey : SurveyInstance [0..1]
add()
update()
delete()
[*]
[*]
[*]
[*]
activeSurveyBeans[*]
surveysBeans[*]
surveyTemplateBeans[*]
surSurveyBeans[*]
ActiveSurveyBean
selectedSurvey : SurveyInstance [0..1]
activeSurvey : SurveyInstance [0..1]
selectedParticipant : Participant [0..1]
activeParticipant : Participant [0..1]
startSurveyInstance()
pauseSurveyInstance()
resumeSurveyInstance()
stopSurveyInstance()
SurveysBean
selectedSurvey : SurveyInstance [0..1]
selectedParticipant : Participant [0..1]
SurveyTemplateBean
newTemplate : SurveyTemplate [0..1]
selectedTemplate : SurveyTemplate [0..1]
question : Question [0..1]
add()
delete()
update()
removeQuestion()
addQuestion()
addOptionToQuestion()
removeOptionFromQuestion()
StartSurveyBean
instance : SurveyInstance [0..1]
createInstance()

Figure 15: Web Client Module's Class Diagram.

ActiveSurveyBean is the core class of this module as it provides the user with the interface of managing a particular experiment/survey and at the same time the GUI is automatically updated to reflect changes in state pushed by devices via socket, web-services, and/or servlets. This is critical for the researcher to control during experiment runtime which devices are: connected, collecting data, delivering sensor data, and the task's status. Having the proper information the researcher might take actions such as pausing or restarting the survey in order to adjust to the state of participant smartphone devices. The GUI presented to the researcher via web browser is an HTML based view generated by *JSF* engine and transmitted via *HTTP* protocol by *FacesServlets*. The interface is organized into three major components: the menu bar, left list view panel and central informative one. Details are depicted in Figure 16.



Active Surveys

(1 of 1) 1

ID	Name	Description	Status
9	instance 1	sadfsadf	CREATED
11	instance 3	from template with now questions	CREATED
15	Survey with tablets		CREATED
14	ssdfsd	sdfasdf	CREATED
13	FSDSAF	DSFASDF	CREATED
12	instance of TEST template	description for instance of test template	CREATED
10	instance with questions	with questions	CREATED
5	experiment 1	description	PAUSED

(1 of 1) 1

Active Survey

Basic Information

Name: instance with questions
Estimated Start Date : 06/04/2014 12:00
Real Start Date :
End Date :
Description * with questions

Participants of the current survey

(1 of 1) 1

ID	Participant	Device	Device Status	DataStatus	Last CMD
143		359861043554759 (Asus)	●	WAITING	EMPTY
144		354793050874660 (samsung)	●	WAITING	EMPTY
103		359959046452433 (Samsung)	●	WAITING	EMPTY
104		355798052551116 (Samsung)	●	WAITING	EMPTY
133		358148043042410 (Sony ericson)	●	WAITING	EMPTY

(1 of 1) 1

Figure 16: A Base User-Interface Page.

5.3 Application Server

Application server provides the major gateway for accessing the web-client application over the intranet or internet. There are a moderate number of open-source modern application servers that provide JSF support for our web application such as: *Glassfish*, *JBoss*, *Jetty,* and *Tomcat*. We validated each application server based on four major characteristics: cross-platform, JSF & Servlet support, JEE support, and administration GUI which are compared in Table 2. All servers express a high performance level and arguably any of them might appear superior under certain conditions. Therefore, the key aspect for selection of the application server is the multi-platform support, J2EE applications and easiness in administration. Glassfish is significantly superior to the rest of application servers. It offers a full-blown J2EE support including the latest versions of JSF, EJB, and *FacesServlets* which are critical components of our web client application. Moreover Glassfish provides a fully functional GUI which simplifies the management process of applications.

Table 2: Application Servers Comparison.

	Cross-platform	**JSF & Servlet Support**	**JEE Support**	**Administration GUI**
Glassfish	***Support***	***All Versions***	***All Versions***	***Advanced***
Jboss	Support	Most Versions	Most Versions	Basic
Jetty	Support	All Versions	Most Versions	Basic
TomEE (Tomcat)	Support	Most Versions	Most Versions	Advanced

5.4 Client Side Implementation Details

This section introduces the implementation details of the client component of the framework which consists of an *Android* application targeting smartphone devices from *Android version 2.3* and above. A simplified but all-inclusive design of the mobile application is depicted in Figure 17. The components are grouped according to *Android's* major building blocks into: *CommunicationHandler, UserGUI, SensorHandler*, and *Core Objects*.

5.4.1 Core Objects

In order to simplify the development process and provide some common ground regarding the design of application classes, two core objects were introduced: the interface *AnalysisObject* and its core implementation *BasicAnalysisObject*. They introduce the simplest object unit within the boundaries of the *Android* application. The interface defines the ability to listen and notify listeners for event triggered during the application runtime. Meanwhile the *BaseAnalysisObject* defines a basic implementation of the functionalities highlighted by the interface. All core classes such as services, sensors and data handlers, as it will be described in the following sections, are specialized objects based on extending the *BaseAnalysisObject*. These objects are depicted in Figure 18(b). Throughout this section the *extend* relationships between *BaseAnalysisObject* and other core classes are not graphically depicted, however all the details will be provided while analyzing them on an item per item basis.

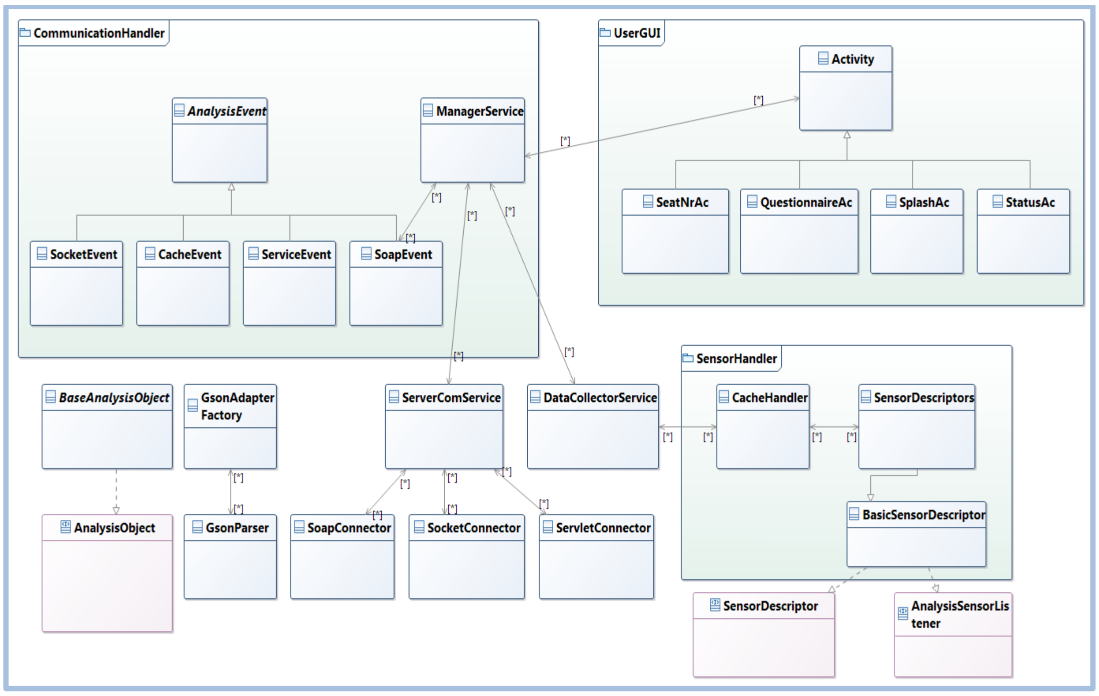
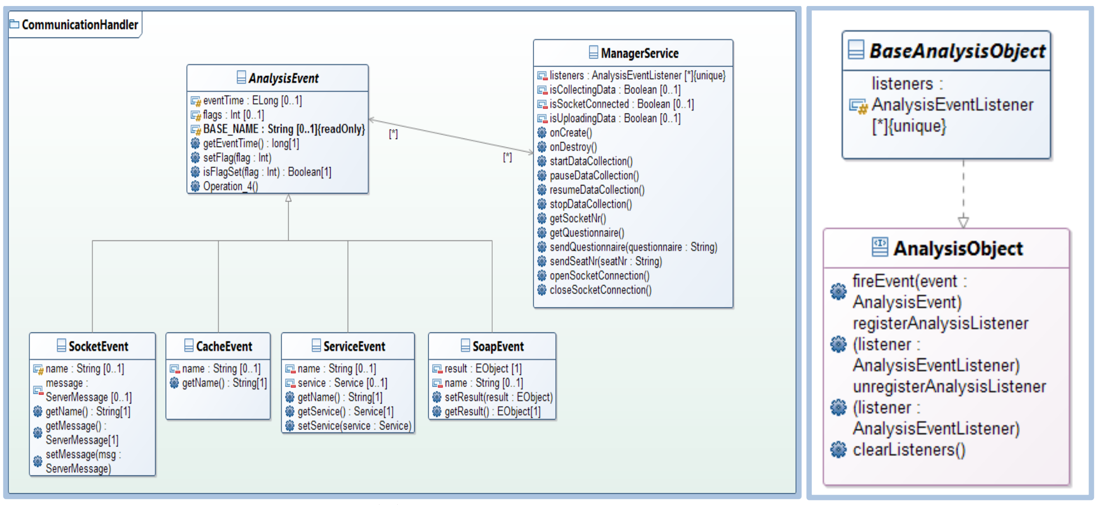


Figure 17: Android Application Class Diagram.

5.4.2 Communication Handler

CommunicationHandler as a module is designed to guide the communication traffic between user interface components and lower level application components. Its detailed class diagram is depicted in Figure 18 (a).


CommunicationHandler
AnalysisEvent
eventTime : ELong [0..1]
flags : Int [0..1]
BASE_NAME : String [0..1]{readOnly}
getEventTime() : long[1]
setFlag(flag : Int)
isFlagSet(flag : Int) : Boolean[1]
Operation_4()
[*]
[*]
ManagerService
listeners : AnalysisEventListener [*]{unique}
isCollectingData : Boolean [0..1]
isSocketConnected : Boolean [0..1]
isUploadingData : Boolean [0..1]
onCreate()
onDestroy()
startDataCollection()
pauseDataCollection()
resumeDataCollection()
stopDataCollection()
getSocketNr()
getQuestionnaire()
sendQuestionnaire(questionnaire : String)
sendSeatNr(seatNr : String)
openSocketConnection()
closeSocketConnection()
SocketEvent
name : String [0..1]
message : ServerMessage [0..1]
getName() : String[1]
getMessage() : ServerMessage[1]
setMessage(msg : ServerMessage)
CacheEvent
name : String [0..1]
getName() : String[1]
ServiceEvent
name : String [0..1]
service : Service [0..1]
getName() : String[1]
getService() : Service[1]
setService(service : Service)
SoapEvent
result : EObject [1]
name : String [0..1]
setResult(result : EObject)
getResult() : EObject[1]
BaseAnalysisObject
listeners : AnalysisEventListener [*]{unique}
AnalysisObject
fireEvent(event : AnalysisEvent)
registerAnalysisListener (listener : AnalysisEventListener)
unregisterAnalysisListener (listener : AnalysisEventListener)
clearListeners()

(a) (b)

Figure 18: (a) Communication Handler Class Diagram (b) Core Objects.

This module plays an intermediary role of exchanging information among other modules. It serves the user interface components to receive the state of critical components specifically related to server communication and publish the appropriate visual notifications. The core class of the *CommunicationHandler* is the *ManagerService* designed as an *Android Service*. Its primarily objective consists of handling user and server requests and fire events indicating application state updates. User requests refer to intents provided by smartphone users via the available GUI meanwhile the server requests are delegated *ServerComService* module. *ManagerService*'s core functionalities include: establishing and managing socket connection, handling sensors data collection and delivery, and sending SOAP requests to web server for accessing survey related information, providing answer to questionnaires, and smartphone's status related data.

5.4.3 Sensor Handler

Sensor data are managed by different components from the moment of acquisition until the delivery of file data structure which include: *Sensor Listener, Sensor Manager, Sensor Cache Mechanism, Data Storage Handler,* and *Data Uploader*. Our implementation *Sensor Handler* module encapsulates the above mentioned components except the *Data Uploader*. *Data Uploader* was excluded from this module due to its network related nature and as such it is an extension of *Android Service* component. Figure 19 (a) depicts core classes of sensor handler's module and the relations among them. Data records generated by mobile sensors are initially made available to the *AnalysisSensorListener* class and encapsulated onto the *SensorDescriptor* with sensor attributes such as name, data format, and the format of the destination file storage. All sensor records captured by listeners are delegated to *CacheManager* class which provides caching mechanism which on a background thread periodically distributes the cached records to permanent storage files in the selected format. *CacheManager* acquires commands from the *DataCollectorService* and for this purpose provides an interface which includes methods: *start(), stop(), pause(),* and *resume()* the data collecting process.

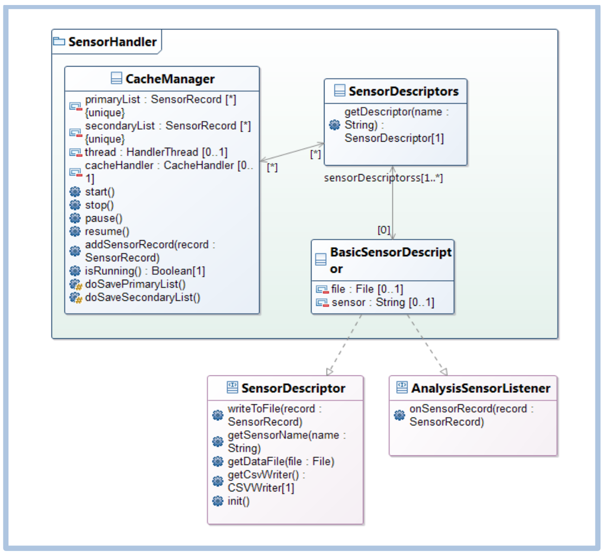
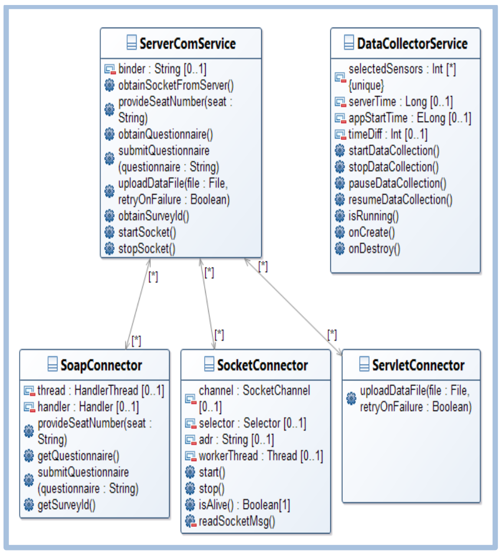
5.4.4 Server Communication Components

One major group of components defined by the framework include server networking related objects: *Socket Client Module, SOAP Web-Service Module,* and *Data Uploader* as described in Figure 19 (b). For this purpose on the implementation design were identified three connector objects: *SoapConnector, SocketConnector,* and *ServletConnector*, responsible for establishing and managing the communication with the server side and providing separate threads for handling respectively soap web-service communication, socket connection and servlet large data file exchange.

5.4.5 GUI Module

The adopted opportunistic approach highlights the necessity of minimizing user interaction with the smartphone device throughout the experiment's duration leading to a simplistic design of the *Android* application. The user interface is composed of four *Android Activities*: *LocationAc*, *QuestionnaireAc*, *SplashAc*, and *StatusAc*, which provide graphical interface to smartphone users. As described in Figure 20 the activities' lifecycle is strictly defined in order to minimize the memory usage and provide the user with only the necessary information. Moreover activities are executed in a predefined queue: *SplashAc* -> *LocationAc* -> *QuestionnaireAc* -> *StatusAc*.

- *SplashAc* checks the initial state of the device in order to establish its suitability for the experiment participation. It audits the *Airplane Mode* setting of the smartphone device, battery level and network connection. If any of these preconditions is not met the device cannot proceed with the experiment. The use is not required to interoperate with the application however the he is notified for the state of the device.


SensorHandler
CacheManager
primaryList : SensorRecord [*] {unique}
secondaryList : SensorRecord [*] {unique}
thread : HandlerThread [0..1]
cacheHandler : CacheHandler [0..1]
start()
stop()
pause()
resume()
addSensorRecord(record : SensorRecord)
isRunning() : Boolean[1]
doSavePrimaryList()
doSaveSecondaryList()
SensorDescriptors
getDescriptor(name : String) : SensorDescriptor[1]
[*]
[*]
sensorDescriptorss[1..*]
[0]
BasicSensorDescriptor
file : File [0..1]
sensor : String [0..1]
SensorDescriptor
writeToFile(record : SensorRecord)
getSensorName(name : String)
getDataFile(file : File)
getCsvWriter() : CSVWriter[1]
init()
AnalysisSensorListener
onSensorRecord(record : SensorRecord)


(a) **(b)**

Figure 19: (a) Sensor Handler and (b)Server Communication Class Diagrams.

- *LocationAc* is the only activity that requires user interaction including the user's exact location specification during the experiment. This information is further used during data analysis to map the sensor data acquired with the physical location of the device.

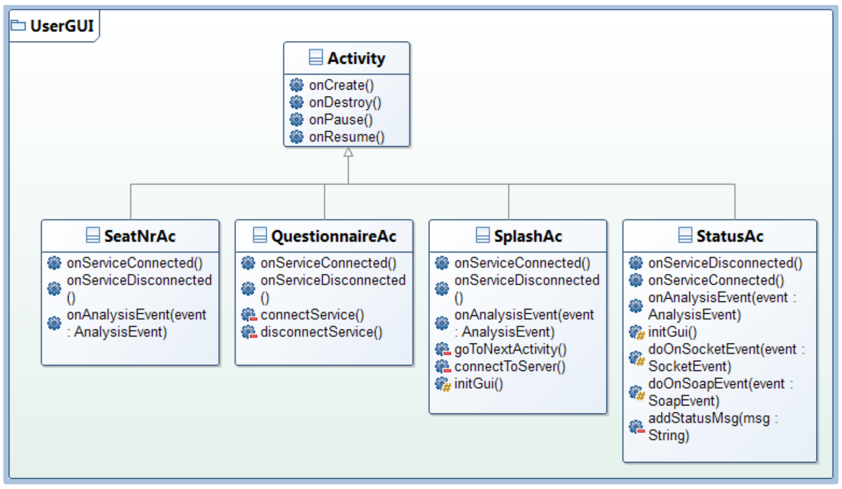


Figure 20: GUI Module Class Diagram.

- *QuestionnaireAc* on the other side is an optional activity. Its execution is based on the server side requirements. Surveys are designed to optionally include questionnaires for gathering statistical information about participants. This activity maps the JSON object acquired by the server side onto a well-designed graphical interface.
- *StatusAc* is the only active activity that provides general information regarding the message exchange with the server side. The purpose of this activity is strictly informatory and no user interaction is required whatsoever. Secondly considering the sensitive private nature of the data received by our framework we intend to comfort participants and keep them updated for every single operation that our application executes, the messages it exchanges with the server, and the scope and size of recorded sensor data.

CHAPTER 6

SURVEYS AND PERFORMANCE TEST ANALYSIS

The mobile sensing nature of the framework implies some performance constraints: the data collection needs to be done without losing any samples, in a high data sampling rate and from multiple sensors. The application has to be able to satisfy these constraints by not having a significant negative impact on the battery life and overall performance of the mobile device. For this purpose we have designed a set of comparative performance tests against a leading benchmark framework of mobile sensing: FUNF.

The second group of tests has been designed to evaluate the entire functionality of the framework during real life experiments. This includes the ability of the framework to synchronously collect sensor data from multiple participants and the analysis of the potential use of these sensor data for audience emotion recognition. We took advantage of the live broadcast of the *FIFA World Cup 2014 in Brazil* in order to organize two social experiments attended by a group of volunteers.

This chapter includes the test setup environment for performance comparisons, analysis of the acquired data and highlights the important improvements our framework introduces in mobile sensing for social experiments. Also, a detailed description is included regarding the organized social experiments, their general characteristics, setup, duration and amount of data received. Based on the results we have evaluated our framework’s approach toward social experiments. We have highlighted issues arising with real experiments and improved the system based on the received feedback and experiences gained during the experiments.

6.1 Performance Testing

Major mobile sensing frameworks include server and client applications. Commonly it is accepted that the mobile applications are significantly more vulnerable to performance related risks due to the limited resources that smartphone devices provide. Server applications on the other side run on powerful machines and performance risks are noticeably lower. Our *Android* application is designed to access the maximum number of sensors available on the smartphone device. In this context the main challenge arising is ensuring the data collection quality using the variety of: sensors and specification setups that the mobile market currently offers. Providing performance tests on each possible configuration setup is not a feasible approach. Therefore, we have categorized smartphone devices based on their performance indicators into three groups: low end, mid end, and high end devices and perform our application tests for each device category. Considering our framework's characteristics the *Android* application puts a particular stress on: IO architecture, CPU utilization, and battery capacity. Multiple mobile sensors are capable of generating hundreds of samples per second. Processing this high rate of data from acquisition to caching and permanent storage, heavily involves IO operations stressing both: storage capacity and operations' speed. Even though the application does not involve considerable amount of mathematical operations, it runs on a multithreaded environment and multiple threads (socket thread, sensor data collection thread, SOAP thread, and data delivery thread) are constantly active. Therefore, maximum CPU utilization is important in handling without delays the tasks defined by each thread. Finally, the limited battery life is a major constraint for smartphone devices specifically under excessive usage of resources such as mobile sensors. It is critical to benchmark devices and identify the physical limitations under the battery life aspect.

Based on these criteria we evaluate the smartphone devices in the three previously defined categories. We have intentionally omitted other critical performance indicators such as 2D and 3D rendering speed of GPU hardware acceleration, as they are not utilized within our *Android* application.

6.1.1 Device Categorization

Based on our benchmarking categories we conducted several tests on different *Android* devices. Our primary objective is to rank the smartphone devices into our low, mid, and high end categories and therefore estimate the correct performance gain of our framework's mobile sensing component against the state-of-the-art framework FUNF. We conducted the benchmark process on a number of Android smartphones: *Samsung Galaxy S4*, *Samsung Galaxy S2*, and *Sony Ericsson xPeria NEO*. On each of these devices we executed a wide range of hardware tests related to the CPU performance, IO operation speed and battery capacity. Performance tests were conducted by using two third party applications: *Linpack* for testing CPU performance and *AnTuTu* for testing IO operations.

Table 3: Smartphone Devices Performance Categorization.

	CPU Result (*cpu*)	IO Result (*io*)	Battery Result (*bat*)	Memory Result (*mem*)	Final Result	Category
Samsung S4	0.74	0.73	0.90	1	0.79	**High**
Samsung Galaxy S2	0.48	0.45	0.65	0.75	0.53	**Mid**
Sony Ericsson xPeria	0.23	0.16	0.6	0.5	0.31	**Low**

Battery and SD card capacity values were collected from the devices' specifications. Testing procedure was established in the following way: for each device we run ten times the entire package of tests included in the *Linpack* and *AnTuTu* applications and the final result stands for the arithmetic average of the obtained performance values. In order to address the differences in scaling level, the final result was normalized to a *[0;1]* scale. *Zero* represents the lowest possible score while *one* stands for the highest score achieved by the best performing smartphone device. Devices tested in Table 3 were evaluated against the *HTC One* device which reflects not-normalized results of 1864 for CPU speed, 6775 for the IO performance tests, 128 GB for SD card memory, and approximately 20 hours of battery duration during *talk time* conditions.

The normalization was implemented against these results and the output is represented respectively in *CPU Result*, *IO Result*, *memory result*, and *battery result* columns of the Table 3. The final result (see Table 3 *'Final Result'* column) stands for the combined value of the four obtained parameters normalized on a *[0;1]* scale by multiplying the with weight factors. We have assigned CPU and IO weight factor of *0.35* as they are highly critical and significantly impact the overall performance of the mobile application, meanwhile battery has been assigned the *0.2* weight value and *0.1* for the *SD Card* capacity. The final result (*R*) is calculated as follows:

$$\boxed{R = cpu \times 0.35 + io \times 0.35 + bat \times 0.20 + mem \times 0.10}$$

6.1.2 Application Test Setup

The overall performance of the mobile component of our framework is measured based on the core parameters regarding per heap memory usage, the number of threads, the number of objects created during runtime, GC calls (*Garbage Collector*), application response time, and number of ANR occurrences. These parameters are designed to quantify the overall picture of the application and do not intend to highlight low level performance indicators. Our core concern is related with the participant's user experience as the mobile sensing application ought not to degrade the overall system performance. From this perspective, CPU usage and heap memory allocation play a critical role as their drastic usage effects the lifecycle of other applications on the participant's device. Avoiding this scenario is fundamental for attracting volunteers on participating on social experiments. For the purpose of comparing the *Android* mobile component of our framework, apart from the application developed within the scope of our proposed framework we developed a second application based on FUNF framework. In order to create as homogeneous environment as possible for achieving valid and reliable results we run the test on the same application stub by modifying only *Sensor Manager* and *Sensor Listener* modules. Each application is installed on three Android smartphones, one per each device category. Tests were designed to provide results for different durations and sensors' setups.

6.1.3 Application Test Evaluation

Overall we executed three tests based on different configuration parameters as described in Table 4, on each device category from Table 3.

Table 4: Test Configurations for Framework Performance Evaluation.

Test Nr.	Duration (min)	Nr. of Active Sensors
1	15	4
2	30	6
3	45	7

During execution time each device was connected to the *Eclipse IDE* for monitoring the performance parameters: *heap size, running threads, GC calls*, and *number of created objects*. Initially all other applications, except *Android* services, were inactive. During the experiment we randomly interact with the smartphone in order to visually evaluate the impact of the application on the overall performance of OS and detect any possible ANR events. For the last experiment additionally were measured: the amount of collected data and the battery consumption level. Obtained results for each experiment are respectively illustrated in Table 5, Table 6, and Table 7. Test results indicate significant performance gains of our framework's mobile component (MC) against FUNF. The most critical improvement to be highlighted is the total avoidance of ANR events which would have very negative effect had they occurred during real experiments. Meanwhile in case of FUNF application, ANR events are primarily met for the mid-end device (*Samsung Galaxy S2*). When number of sensor exceeds 6 the device frequently shows sign of ANR and for 7 sensors it practically becomes non-responsive. High-end device (*Samsung Galaxy S4*) appears to handle better high number of sensors however, for 7-sensor configuration ANRs frequently interrupt the normal flow of data collection. Low-end device (*Sony Ericsson xPeria*) on the other hand is mostly responsive and ANR events rarely appear for the 7-sensor configuration. This can be explained by the poor internal configuration of the device which includes only 4 sensors generating less records compared to 2 other devices. All 3 executed tests indicate significant improvements for all performance parameters. The heap memory is highly stressed by FUNF however it was progressively improved by our framework. For the mid-end device and 7-sensor configuration (Table 7) the heap memory used is reduced by a factor of 2.4. An important reduction of the overall number of objects is visible specifically for long-lasting experiments (Table 6 and Table 7), for mid-end and high-end devices reaching a reduction factor of 5.5.

Table 5: Results-Test No.1

	Heap Size (MB)		**Number of Threads**		**GC Calls (per second)**		**Max Nr of Objects (x1000)**		**ANR**	
	MC	FUNF	MC	FUNF	MC	FUNF	MC	FUNF	MC	FUNF
Galaxy S4	16	17	2	7	0.35	2.3	65	128	No	No
Galaxy S2	9.5	15	2	7	0.9	4.5	60	120	No	Rare
xPeria Neo	9.2	12	2	7	0.3	3.7	51	113	No	No

The limited number of created java objects, reflects a significant improvement, by a factor of 7-10, of the GC's interference to the ordinary flow of the application's runtime. The conservative approach regarding thread creation improves overall performance from both: memory and CPU perspective , ultimately reducing the battery consumption by *40-60%* as it is depicted in Table 7 which is a closer approximation of a real experiment duration. The last experiment's data provide the first real indicator of the size of sensor data collected on a 45-mins lasting survey. From a single device we accumulated up to 28.5 MB of sensor data.

Table 6: Results-Test No.2

	Heap Size (MB)		**Number of Threads**		**GC Calls (per second)**		**Max Nr of Objects (x1000)**		**ANR**	
	MC	FUNF	MC	FUNF	MC	FUNF	MC	FUNF	MC	FUNF
Galaxy S4	17	23	2	9	0.4	3.3	68	299	No	No
Galaxy S2	10	18	2	9	1.3	8.1	63	335	No	Freq.
xPeria Neo	9.3	10.7	2	9	0.4	3.8	57	149	No	No

The performance improvements introduced in the previous paragraph are attributed to the adaptation of our framework to the limited resource environment available on smartphone devices. In contrast to FUNF, we intentionally provided one single thread to handle the data generated from all the sensors in order to minimize very frequent context-switches among available threads. The architecture of FUNF is designed to provide the sensor data in a unified JSON format leading to frequent creation/parsing of string objects which cause the GC to interfere up to 8 times per second. This is a major factor that causes delays in application response time because GC blocks other threads while allocating/deallocating memory and its execution time is non-deterministic at the application level.

Table 7: Results-Test No.3

	Heap Size (MB)		**Number of Threads**		**GC Calls (per second)**		**Max Nr of Objects (x1000)**		**Battery Cons. (%)**		**Data size (MB)**	**ANR**	
	MC	FUNF	MC	FUNF	MC	FUNF	MC	FUNF	MC	FUNF	FUNF & MC	MC	FUNF
Galaxy S4	18	26	3	11	0.4	3.5	101	345	9	21	28.5	No	Freq.
Galaxy S2	11	27	3	11	1.4	8.6	94	572	14	29	20	No	Fail
xPeria Neo	9.8	15.5	3	11	0.7	4.4	72	212	4	11	13.6	No	Rare

Moreover, we evaluated the official FUNF's test application from the *Google Play* [63] [64] and we encountered the same performance issues we identified in the previous paragraph. Collecting data from more than four sensors significantly interferes with the overall user's experience with the device and the application crashes frequently interrupting the sensor data collection tasks. Therefore we may conclude that our framework provides a feasible way for multiple-sensor data collecting tasks for social experiments. Its design toward efficiency reduces the amount of resources required by the mobile application. The limited battery consumption may allow the execution of long lasting experiments without interfering on the user response time of the smartphone.

6.2 Surveys Evaluation

The performance results obtained in the previous sections highlight the readiness of our framework to be applied on real emotion analysis experiments. Mobile component's performance parameters overall indicate that the issues commonly experienced in mobile sensing platforms such as: ANR and very rapid battery consumption, are properly addressed for all the three smartphone categories. Furthermore the heavy access on platform's sensors does not interfere with the overall functioning of the system and specifically the background data collection is virtually unnoticed by the participant.

Therefore, in this section we organized two major experiments with sport fans. Taking advantage of the ongoing *FIFA World Cup 2014* we set up our surveys for two football matches of the tournament. Below we provide the detailed description of the experiments' organizational efforts, the overall data collection and delivery process, the obtained results and evaluation for probable emotion analysis purpose.

6.2.1 Survey Setup

In the context of our framework we organized two experimental sport based events. Each of them was established around a particular event of the *FIFA World Cup 14* specifically on *19 June 2014* during *Uruguay – England* and on *04 July 2014* during

France – Germany matches. The organization process for both events was organized in the following manner:

Two to three days prior to the event we notified via e-mail and social networks potential volunteering participants with the catchy phrase "*World Cup! Let's watch together*". Within our notification list we included a heterogeneous audience from different social backgrounds and nationalities. The day of the experiment considering the number of volunteered participants we setup the environment in one of the *Informatics Institute's* auditorium of the *Middle East Technical University*. Prior to the experiment we distributed via e-mail the mobile application for installation and a handful of requirements which we estimated would take up to 30 minutes from participants' time to read, understand and complete if any task is required. These requirements are summarized as follows:

- Participants are requested to carry with them *Android* smartphones possibly fully charged.
- Prior to the experiment we request the installation of a 2MB *Android* application which will serve to collect sensor data anonymously.
- During the experiment we invite all participants to activate the *Airplane Mode* setting of the smartphone in order to avoid any disturbance occurring during the survey.
- During the experiment it is important to hold the smartphone in the chest pocket of your shirt.
- Prior to the experiments we request device's identification number which will be used only for mapping participating smartphones with the acquired sensor data.
- Optionally statistical surveys will be requested for submission not requiring any private data whatsoever.

The auditorium where the experiment would take place was equipped with a projector, two laptop computers, and a camera. The projector was used to display on a large board the football match transmitted online by one of the laptop computers. In the second computer an instance of the server component framework was running in order to provide the control station of the researcher. Both machines were running on a *Windows* environment however we performed a wide range of tests on both *Linux* and *Windows* machines particularly during development phase. Additionally we created an intranet for establishing the connection between server and mobile client devices. Only registered devices were eligible for experiment participation and considering the closed network established, we did not activate the encryption on any of the connection methods used to exchange messages between the server and the client applications. At the same time we positioned a camera toward participants and recorded the entire experiment as depicted in Figure 21.

For the first survey we targeted 10 participants and for second survey 15. The experiment was planned to be executed in two parts with a break during the 15-minutes half-time break. During this period devices are instructed by the researcher to stop collecting data and deliver the data acquired up to that moment. The main purpose for this approach is to demonstrate the flexibility of the framework for data collection from multi-parts experiment which at the same time reflects an important side effect: optimized battery and memory consumption. At the end of the experiment, the second part of the data is delivered. Data are organized in separate CSV files for each sensor and delivered to the server side in the same format. Our framework targets primarily users' *Android* smartphones however; we arranged additionally three spare devices in case of any failure during the experiment or for registered participants who actually did not possess an *Android* smartphone.

6.2.2 Overall Framework Performance

Both experiments were executed successfully and significant amount of data were gathered from smartphone devices of the participants. In the first experiment we targeted 10 participants however we were able to attract only seven and we utilized two of our three spare devices. Notification to the targeted participants was sent one day ahead and around 10% of the notified people responded positively for the experiment participation. The experiment lasted around two hours and we were able to collect sensor data for a period of 75 minutes. We were unable to monitor the first 15 minutes as participants did not appear on time for installation of the *Android* application and smartphone registration procedures. The total size of the collected data throughout the experiment was 360 MB.

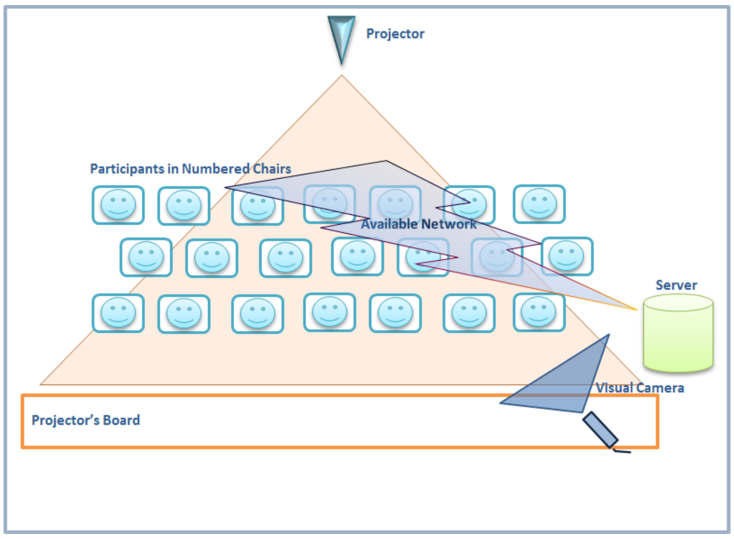


Figure 21: Environment Setup of Emotion Recognition Experiments.

For the second experiment we targeted a wider audience of 15 participants, therefore the notification period was extended to three days prior to the experiment's day. This time confirmed participants were instructed via e-mail specifically to install the data collecting application and have their smartphones fully charged. Around 15% of the notified people (16 participants) responded positively to the survey request. The experiment's duration was practically identical to the first one and we were able to collect data for 78-minutes period. During this survey we experienced an internet blackout which prevented us of collecting data for the remaining 12 minutes period. In total: from 7 sensors of each participating smartphone we collected 930 MB of data.

Both experiments highlight the usability of our framework in collecting sensor data from multiple sensors in a synchronized fashion during real events. The organizational process was significantly simplified as we were able to organize social experiments within few days. Cost related to the necessary hardware for the experiment was limited to only: two laptop computers, one modem and optionally three spare smartphones. The framework overall performed according to the design requirements. It successfully handled data collection from up to 16 devices from real time experiments and it demonstrated the capability of splitting the experiments, if necessary, into smaller parts. The data synchronization among devices was achieved to a maximum delay time of 200 milliseconds. Data delivery also was performed almost errorless as in the second experiment one of the devices failed to maintain the connection. In that case the collected data were extracted manually from the device's *SD Card*. However, the organization of experiments for real events poses a challenge as unexpected situations may take place as we experienced the absence of internet connection for a period of 12 minutes during the second experiment preventing us from collecting sensor data. Both: server and mobile clients performed according to the requirements and most importantly no ANR events were met during the experiments. This is a major milestone which demonstrated the ability of current hardware specification to handle heavy load of multiple sensor data without effecting the performance of the mobile OS.

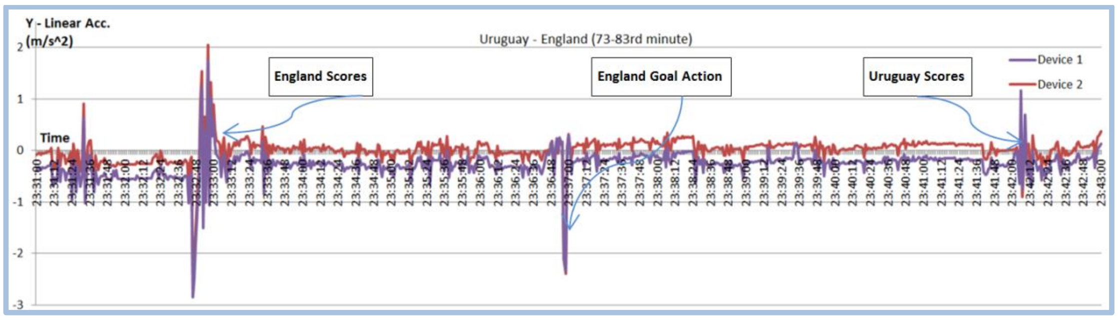


Figure 22: Experiment 1 – Linear Accelerometer Graph on Y-axis for two participating devices between 73rd and 83rd minute.

6.2.3 Data Evaluation

The collected sensor data from both experiments were analyzed in order to evaluate their usability in the context of audience emotion analysis. The analysis process consists of extracting the collected data, displaying in a graph format and matching the records' timestamps with particular football match events from the recorded videos. Our analysis indicates a correlation between recorded sensor data and the particular events happening during the game which perhaps are sources of high arousal emotional states. Regarding acquired data from the first experiment we extracted particular timeframes: between 73-83rd and 36-42nd minute respectively depicted in Figure 22 and Figure 23, during which relatively more striking events occurred. Figure 22 depicts the recorded data from *Linear Accelerometer* sensor along the Y-axis only. For simplicity we have included only two devices and we were interested in events such as: *Dangerous Action* and *Scored Goal* which correlate with high amplitudes of sensor records indicating high arousal of the emotional state of the audience. Meanwhile, in Figure 23 we highlight the *Accelerometer* sensor data from a single device recorded along all three axes for a timeframe of six minutes and we followed the same event type as in the previous image.

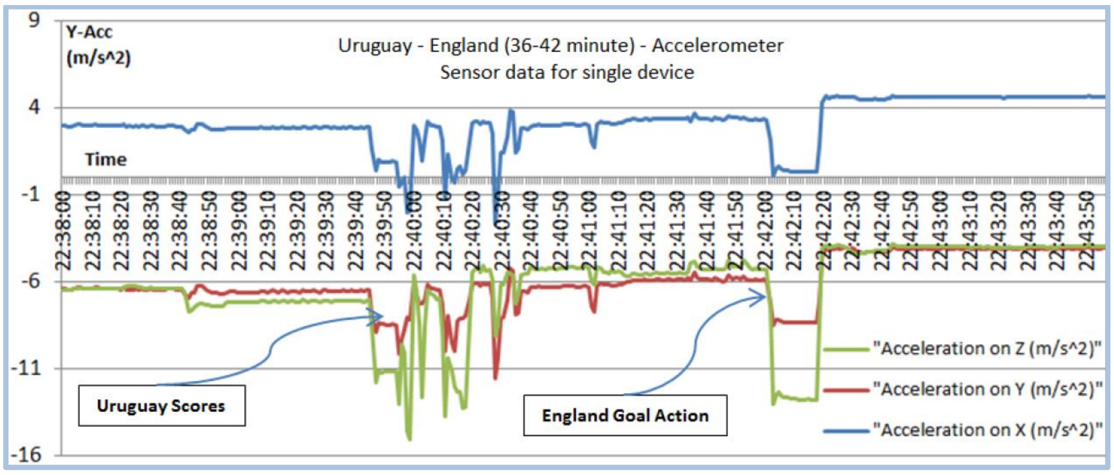


Figure 23: Experiment 1 – Accelerometer Graph for all axes of a single device between 36th and 42nd minute.

Data from all three axes express similar patterns throughout the events during the match (while having different magnitudes). Similar conclusions can be drawn from the collected data of the second experiment which introduces a wider scale of our framework application. This time we included data recorded by the *orientation* sensor in order to highlight the usability of multiple sensors. Figure 24 highlights orientation graphs on *X*, *Y*, and *Z* axes in terms of angle degrees. In line with the previous results, the tagged events correlate with a significant disruption in magnitude of the orientation sensor data over all three axes. Accelerometer data from two devices during a single *goal score* time frame is shown in Figure 25. This figure serves as a good indicator of the correlated nature of the acquired data on different devices during a striking event.

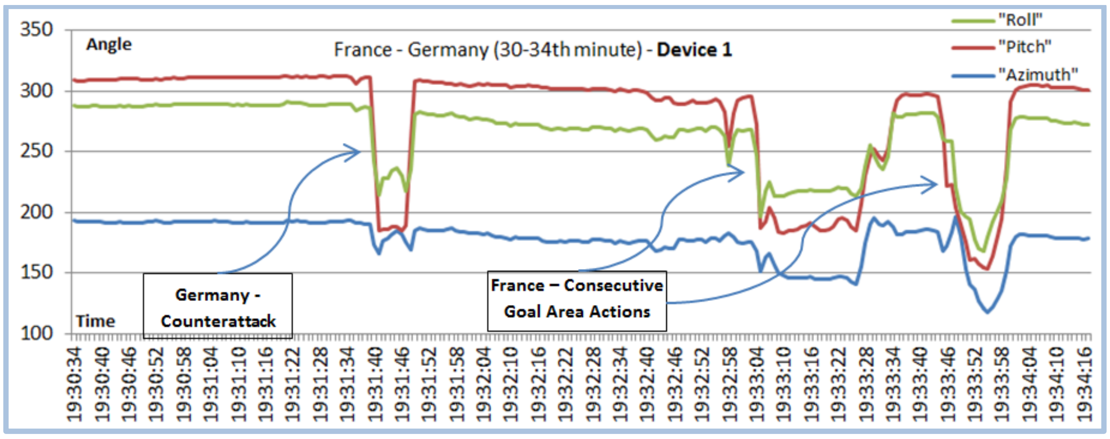


Figure 24: Experiment 2 – Rotation Sensor Graph on all axes for a single device between 30th and 34th minute.

The reaction to the goal event is reflected practically in identical times (23:32:48) in both graphs. However, not all sensors seem to generate relevant information from the emotion analysis perspective. In the Appendix section we have included a large variety of results from both experiments regarding different sensors. Particularly in Figure 26 we have including acquired data from all sensors for a single device. The most relevant sensor is observed to be the *Accelerometer*, generated data of which we extensively used during our analysis process. Recorded data from the microphone on the other side, are more ambiguous and do not provide any clear indication of particular events throughout the observed match. The environmental noise is the primary source of the disruption; therefore its usage in indoor environments ought to be further investigated.

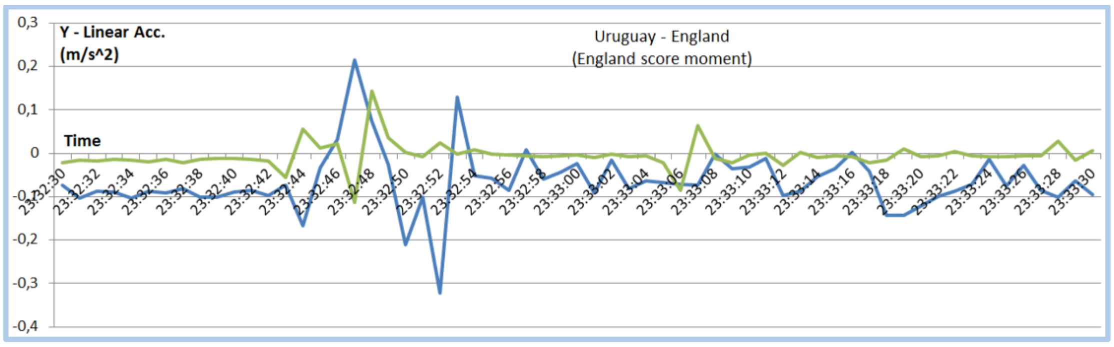


Figure 25: Experiment 1, Accelerometer data for 2 devices on a specific goal timeframe.

A broader picture of the acquired data is depicted in Figure 27 and Figure 28 for the first and second experiment respectively. Overall, a correlation between tagged events and sensor data is observed in multiple devices. However, a more vigorous attempt on data analysis requires the application of more sophisticated data mining techniques, which is beyond the scope of this thesis. Moreover, the data may be studied for other domains as well, especially activity sensing and body movements tracking. In this case the study would focus on developing particular patterns for certain activities and applying classification methodologies on the acquired data.

CHAPTER 7

FUTURE WORK AND CONCLUSIONS

7.1 Conclusions

This study introduces a framework for synchronous data collection from multiple smartphone sensors from a group of people. It is intended to serve across different research domains and industries, and we provided the evidence for its applicability during real life events particularly for audience emotion analysis purposes. Our implementation of the framework expressed high reliability and efficiency on collecting data from multiple sensors simultaneously. We provided performance test results indicating significant improvements against a state of the art mobile sensing framework: FUNF. Our framework was able to collect data from 7 sensors concurrently without degrading the performance of the Android OS of the test devices.

The pilot study organized during FIFA World Cup 2014 used to evaluate our design assumptions and framework's applicability in emotion analysis domain. During this study the framework was able to collect sensor data from multiple participants and deliver them on a remote location on a secure and optimized manner. Our smartphone led approach successfully addressed the time consuming and expensive nature of emotion analysis type of experiments. Within few days we were able to organize experiments on real social events including up to 16 people by using only participants' smartphone devices. Finally, after a close analysis of the acquired data during two experiments, we demonstrated the correlated nature of sensor data and the emotional state of the audience. Overall, the framework's functionalities can be summarized as follows:

- Data collection from multiple sensors simultaneously.
- Data collection from multiple participants in a synchronized fashion.

- Automatic data delivery from all participants to the server’s location.
- Create/update/view experiments including their parameters i.e timing, name and description, and assigned devices.
- Ability to remotely control the flow of the experiment i.e starting and stopping data collection from participating devices.
- Client web application for controlling the status of the experiment and participating devices.
- The ability to design questionnaires to make available to participants on their smartphone devices prior/after the experiment.
- Keep track of all applied experiments and the devices involved in each of them.
- Split a single experiment into smaller parts and remotely control the start/end of each segment.
- Keep track of the commands sent to participating smartphone devices: if they were received and processed.
- Statistical information generated regarding experiments and participants, available through web-services.

It is important to emphasize the restrictions that come with the introduction of smartphone devices and mobile sensing in place of custom made devices. Smartphones encapsulate all mobile sensors in a single package and as a result, can trace only one part of the body at a time. Therefore as we identified the chest pocket as the most suitable location of the smartphone during experiment’s duration we unequivocally removed from the overall picture other body part’s movements such as: arms and legs. Moreover, available smartphone devices introduce a wide range of hardware configuration. The internal set of sensor and the hardware quality differs from device to device, making it practically impossible to apply it on a person base. However, the wide reach of smartphones provide a ready infrastructure for applying larger scale experiments, even globally as the authors of [1] attempt to. The final purpose of the mobile sensing framework presented in this study should not be considered as a replacement of any other methodology or technique applied in emotion analysis or activity sensing domains, but rather as a complementary tool to be quickly applied on a wider scale involving users' smartphone devices.

7.2 Contributions

This study provides important insights for future researchers on the mobile sensing effectiveness and its potential impact on domains such as: emotion analysis and activity sensing. The proposed framework handles all core aspects from sensor data collection, maintenance, and delivery to a remote centralized location. Furthermore, the server side component provides the necessary tools for experiments' results sharing and statistical information generation. The introduced framework is designed for *Android* smartphones and effectively addresses all major available devices from low end to high end categories. All major Android releases, from version 2.2 are supported and successfully tested. The framework minimizes organizational efforts and redirects resources toward data collection and analysis

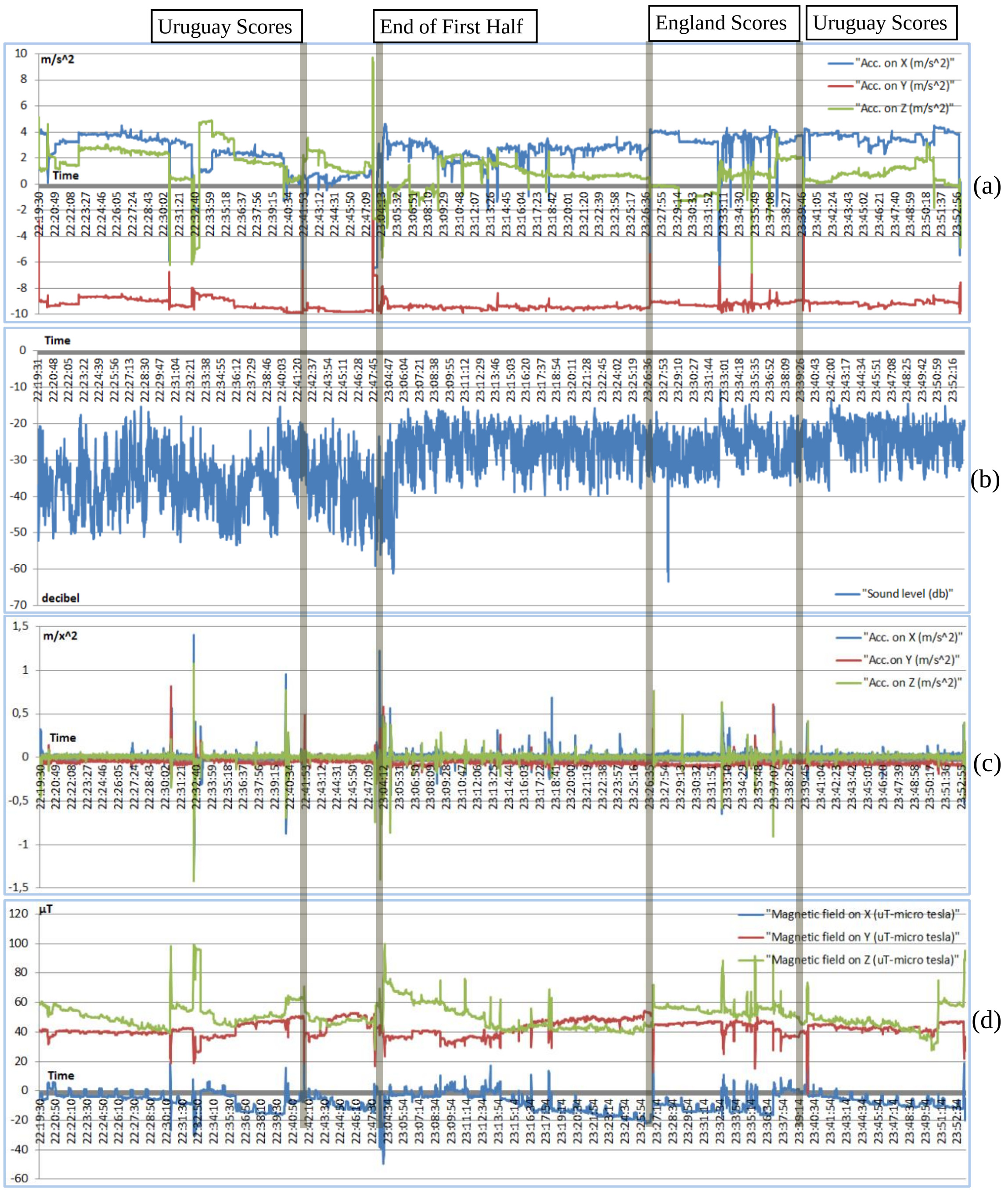
7.3 Future Work

A potential area of research is the adaptation of our framework in other studies specifically regarding domains of activity sensing, body tracking and human-computer. Meanwhile the smartphone industry is still rapidly advancing and the latest devices have incorporated additional sensors: such as biometric sensors. Therefore, integration of these sensors in our framework may introduce a new dimension of data regarding human body state analysis.

Additionally, all modern smartphones include single or double optic camera and recently thermal cameras were also provided as extra on-demand accessories. Collection of data from such sensors may extend the applicability of mobile sensing to other domains. Moreover, extraction and use of other features from the speech data could also be investigated.

Recently introduced *wearable* devices can communicate via *Bluetooth* with smartphones creating an eco-system of smart devices around a person. Wearables such as *Google Glass* and *Smartwatch* may provide extra sensor data regarding head and arm movements which enable more detailed analysis of body gestures and perhaps widen the scope collected data.

APPENDIX


Uruguay Scores
End of First Half
England Scores
Uruguay Scores
m/s^2
Time
"Acc. on X (m/s^2)"
"Acc. on Y (m/s^2)"
"Acc. on Z (m/s^2)"
(a)
Time
decibel
"Sound level (db)"
(b)
m/x^2
Time
"Acc. on X (m/s^2)"
"Acc.on Y (m/s^2)"
"Acc. on Z (m/s^2)"
(c)
µT
Time
"Magnetic field on X (uT-micro tesla)"
"Magnetic field on Y (uT-micro tesla)"
"Magnetic field on Z (uT-micro tesla)"
(d)

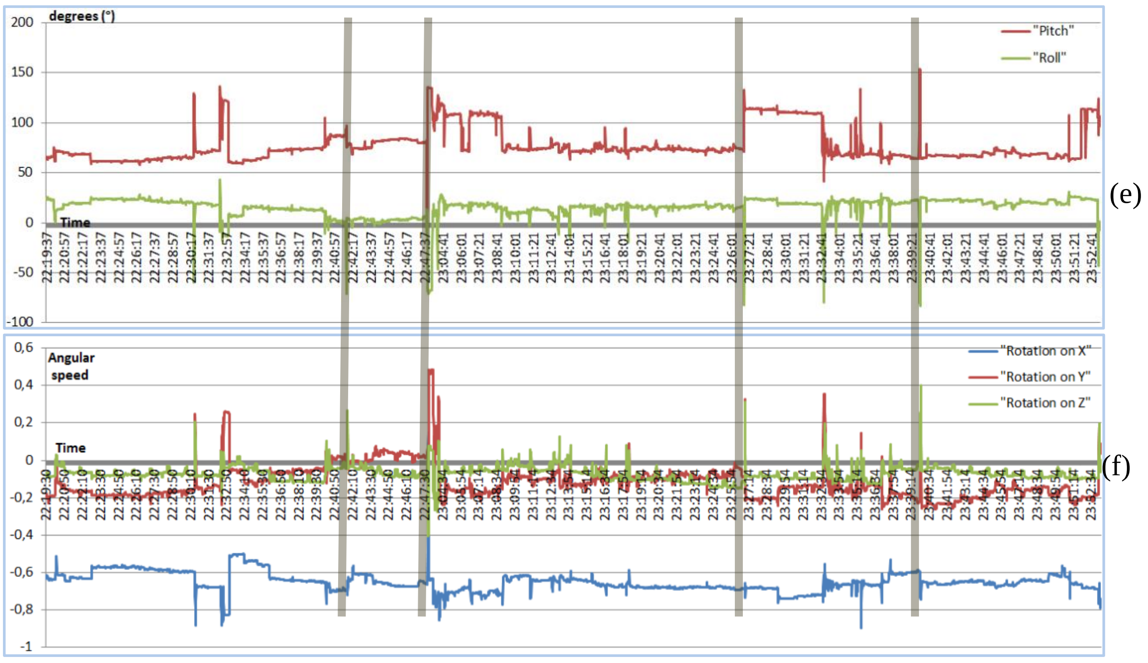
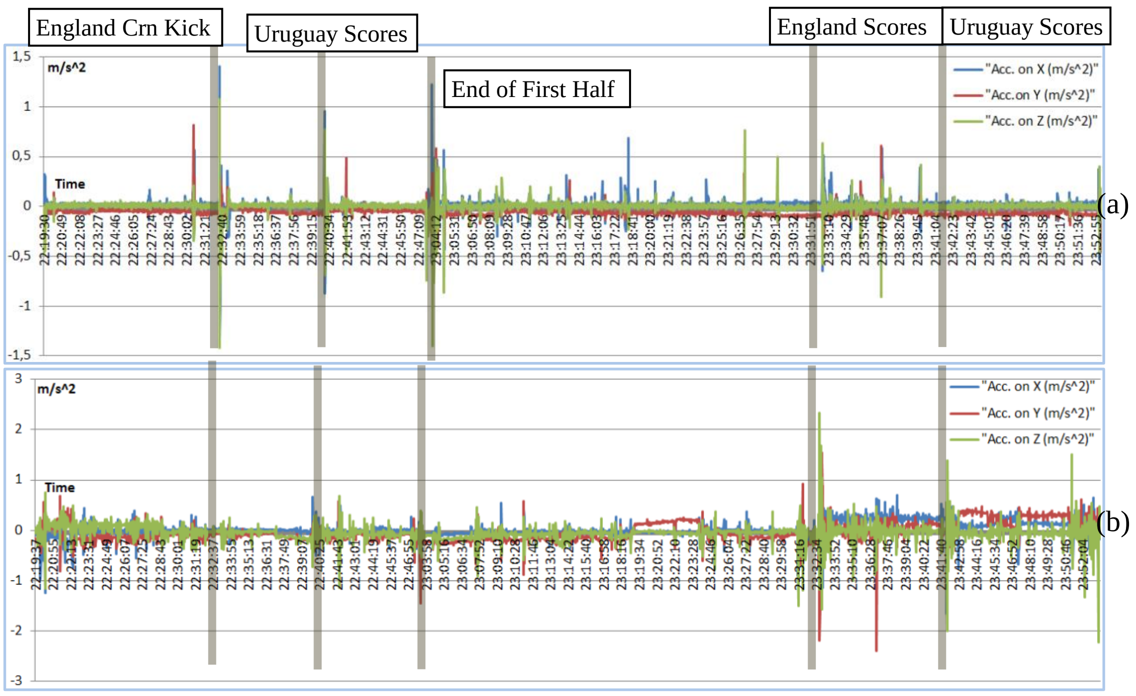


Figure 26: Experiment 1, one single device's sensor data on: (a) accelerometer, (b) sound, (c) linear accelerometer, (d) magnetometer, (e) orientation, and (f) rotation vector.



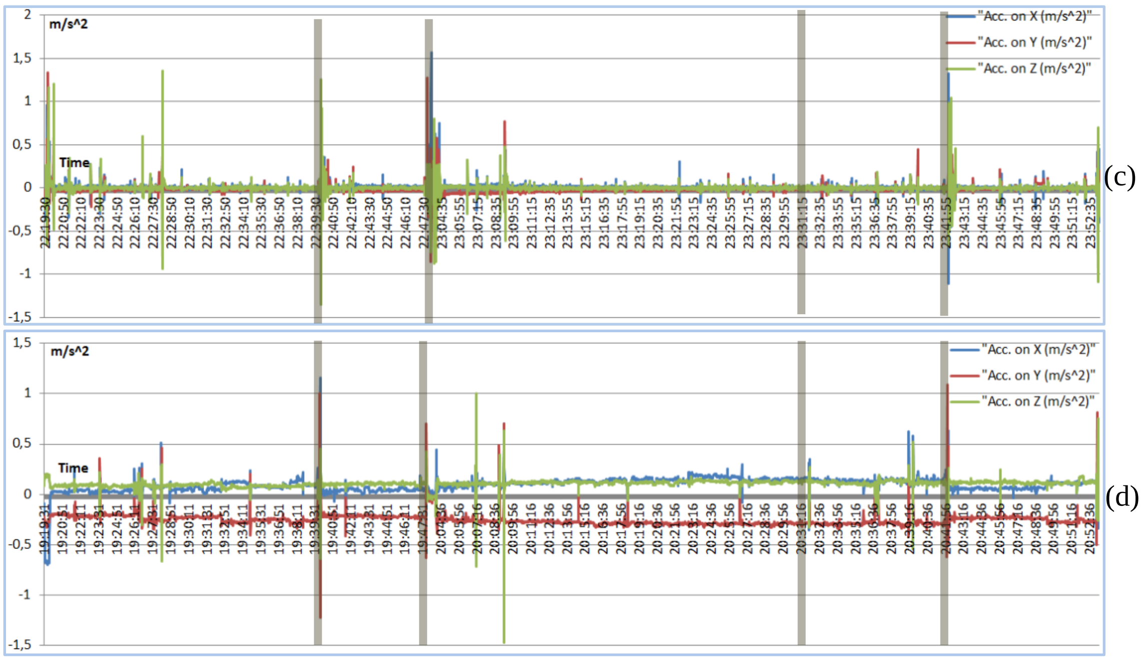
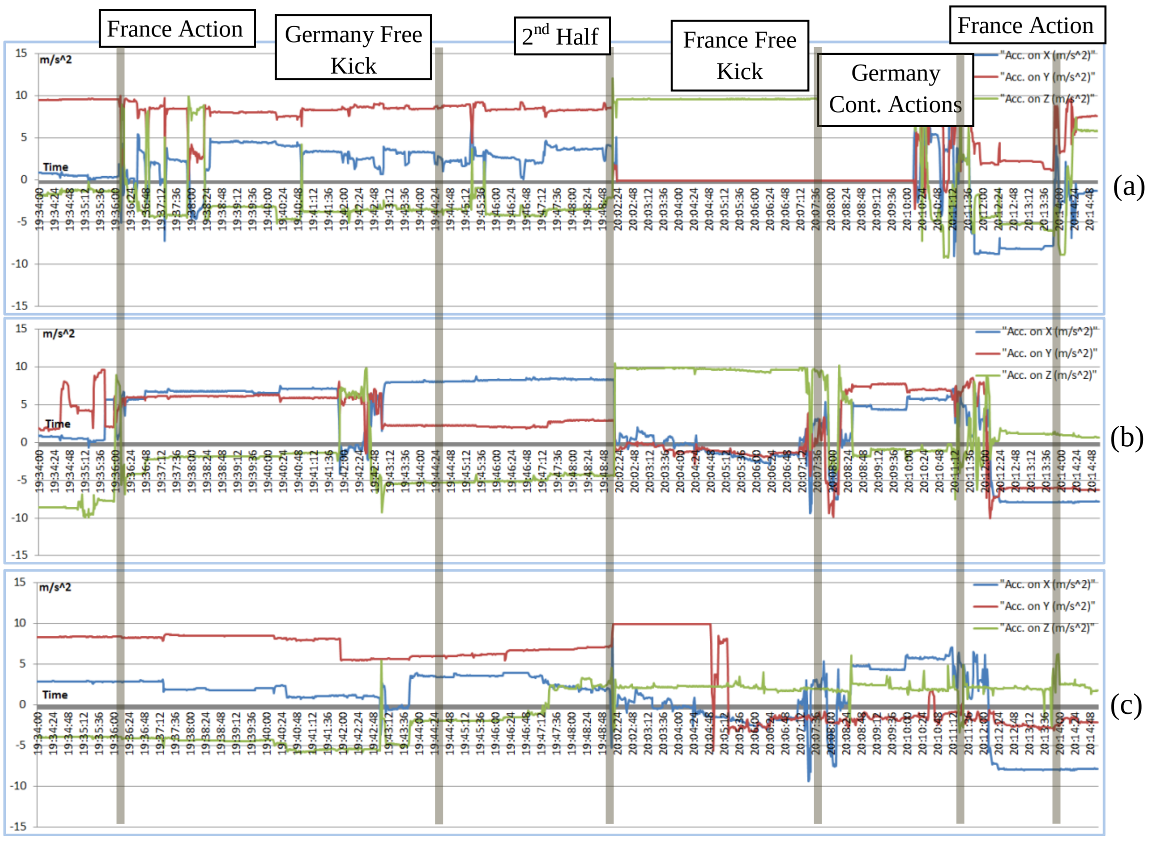


Figure 27: Experiment 1, Accelerometer data for 5 devices: (a), (b), (c) and (d).



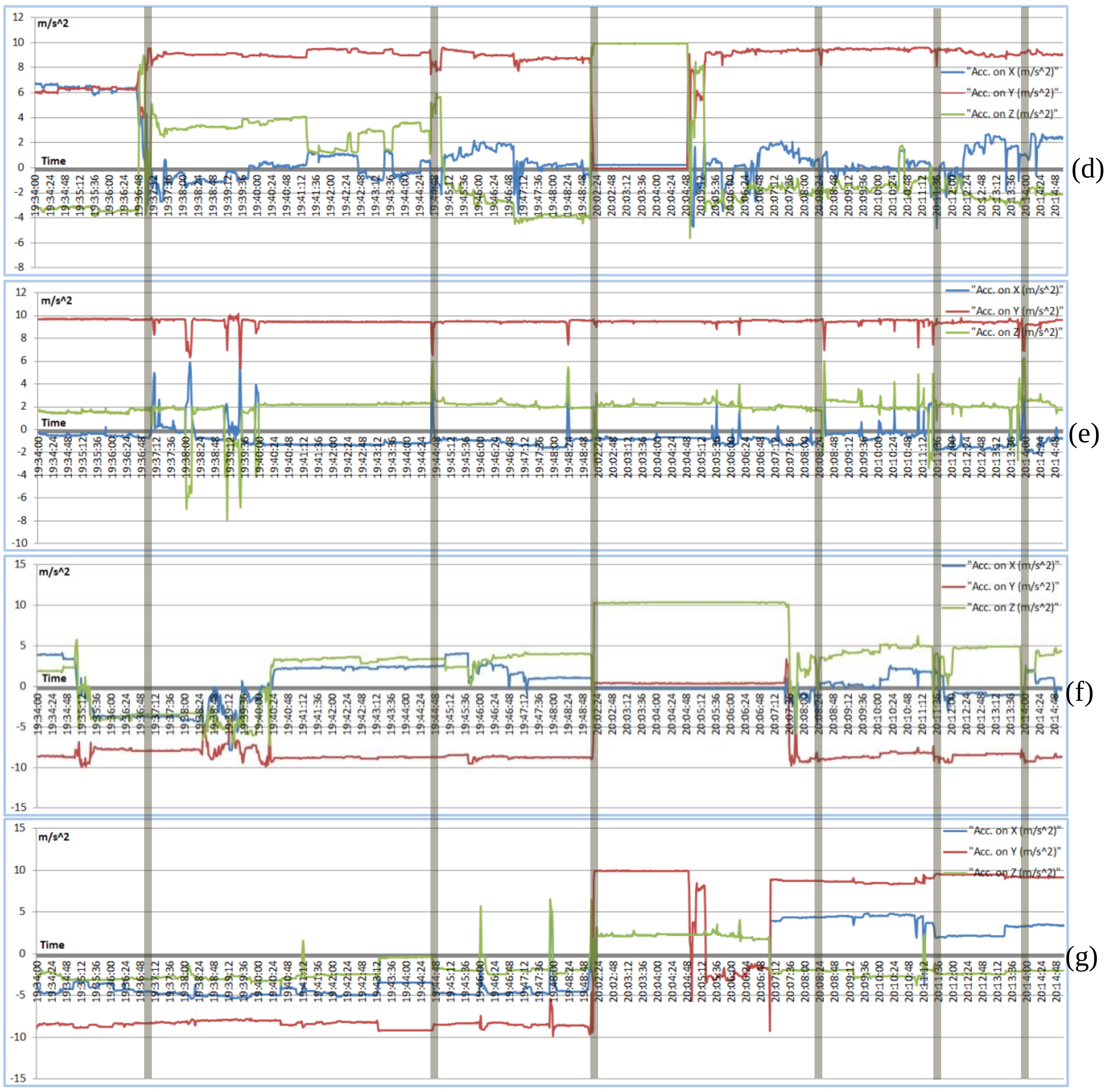


Figure 28: Experiment 2, Accelerometer data between 34th and 60th minute for 7 devices: (a), (b), (c), (d), (e), (f), and (g).

BIBLIOGRAPHY

[1] M. Barbos, E. Pop, H. Lee and L. M. Campos, "UbiPOL: A Platform for Contex-aware Mobile Device Applications in Policy Making," in *UBICOMM*, Lisbon Portugal, 2011.

[2] S. Consolvo, D. W. McDonald, T. Toscos, M. Y. Chen, J. Froehlich, B. Harrison, P. Klasnja, A. LaMarca, L. LeGrand, R. Libby, I. Smith and J. A. Landay, "Activity Sensing in the Wild: A Field Trial of UbiFit Garden," in *ACM SIGCHI Conference*, Florence, 2008.

[3] S. Devarakonda, P. Sevusu, H. Liu, R. Liu, L. Iftode and B. Nath, "Real-time Air Quality Monitoring Through Mobile Sensing in Metropolitan Areas," in *UrbComp*, Illinois USA, 2013.

[4] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo and J. Eriksson, "VTrack: Accurate, Energy-Aware Road Traffic Delay Estimation Using Mobile Phones," in *7th ACM Conference on Embedded Networked Sensor Systems*, New York, USA, 2009.

[5] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng and A. T. Campbell, "Sensing Meets Mobile Social Network: The Desing, Implementation and Evaluation of the CenceMe Application," in *6th ACM SenSys*, North Carolina USA, 2008.

[6] W. Z. Khan, Y. Xiang, M. Y. Aalsalem and Q. Arshad, "Mobile Phone Sensing Systems: A Survey," *Communications Surveys & Tutorials,* vol. 15, no. 1, pp. 402-427, 2013.

[7] L. C. De Silva, T. Miyasato and R. Nakatsu, "Facial Emotion Recognition Using Multi-Modal Information," *Informations, Communications, and Signal Processing*, pp. 397-401, 9-12 September 1997.

[8] C. Busso, Z. Deng, S. Yildirim, M. Bulut, C. M. Lee, A. Kazemzadeh, S. Lee, U. Neumann and S. Narayanan, "Analysis of Emotion Recognition Using Facial Expressions, Speech and Multimodal Information," in *6th International Conference on Multimodal Interfaces*, New York, USA, 2004.

[9] K. Schnindle, L. Van Gool and B. de Geldler, "Recognizing Emotions Expressed by Body Pose: A Biologically Inspired Neural Model," *Neural Networks,* vol. 21, no. 9, pp. 1238-1246, 2008.

[10] G. Castellano, L. Kessous and G. Karidakis, "Emotion Recognition through Multiple Modalities: Face, Body Gesture, Speech," in *Affect and Emotion in Human-Computer Interaction*, Springer Berlin Heidelberg, 2008, pp. 92-103.

[11] S. G. Koolagudi, R. Reddy and S. K. Rao, "Emotion Recognition from Speech Signal Using Epoch Parameters," in *Signal Processing and Communications*, Bangalore, India, 2010.

[12] A. B. Ingale and D. S. Chaudhari, "Speech Emotion Recognition," *International Journal of Soft Computing and Engineering,* vol. 2, no. 1, pp. 235-238, 2012.

[13] M. El Ayadi, M. S. Kamel and F. Karray, "Survey on Speech Emotion Recognition: Features, Classification Schemes, and Databases," *Pattern Recognition,* vol. 44, no. 3, pp. 572-587, 2011.

[14] A. d. S. Sierra, C. S. Avila, J. G. Casanova and G. B. del Pozo, "A Stress-Detection System Based on Physiological Signals and Fuzzy Logic," *Transactions on Industrial Electronics,* vol. 58, no. 10, pp. 4857-4865, 2011.

[15] J. Kim, "Bimodal Emotion Recognition using Speech and Physiological Changes," in *Robust Speech Recognition and Understanding*, I-Tech Education and Publishing, 2007, pp. 265-280.

[16] C. D. Katsis, N. Katertsidis, G. Ganiatsas and D. I. Fotiadis, "Toward Emotion Recognition in Car-Racing Drivers: A Biosignal Processing Approach," *System, Man and Cybernatics,* vol. 38, no. 3, pp. 502-512, 2008.

[17] D. Bernhardt and P. Robinson, "Detecting Emotions From Connected Action Sequences," *Visual Informatics: Bridging Research and Practice,* vol. 5857, pp. 1-11, 2009.

[18] U. Maurer, A. Smailagic, D. Siewiorek and M. Deisher , "Activity Recognition and Monitoring Using Multiple Sensors on Different Body Positions.," in *Workshop on Wearable and Implantable Body Sensor Networks*, Cambridge MA, April 2006.

[19] D. Akimura, Y. Kawahara and T. Asami, "Compressed Sensing Method for Human Activity Sensing Using Mobile Phone Accelerometers.," in *Network Sensing Systems*, Antwerp Belgium, June 2012.

[20] L. Bao and S. Intille, "Activity Recognition from User-Annotated Acceleration Data.," in *PERVASIVE*, April 2004.

[21] M. S. Barlett, G. Littlewort, C. Lainscsek, I. Fasel and J. Movellan, "Machine Learning Methods for Fully Automatic Recognition of Facial Expressions and Facial Actions," *Systems, Man and Cybernetics*, vol. 1, no. IEEE, pp. 592-597, 2004.

[22] R. Cowie, E. Dauglas-Cowie, J. G. Tailor, S. Ioannou, M. Wallace and S. Kollias, "An Intelligent System for Facial Emotion Recognition," in *Multimedia and Expo*, Amsterdam, The Netherlands, 2005.

[23] M. Valstar and M. Pantic, "Fully Automated Facial Action Unit Detection and Temporal Analysis," in *Computer Vision and Pattern Recognition Workshop*, New York, USA, 2006.

[24] P. Ekmap, W. V. Friesen and J. C. Hager, "The Facial Action Coding System: A Technique for the Measurement of Facial Movements," in *Consulting Psychologist*, San Francisco, USA, 2002.

[25] S. Yang and B. Bhanu, "Facial Expression Recognition Using Avatar Image," in *Automatic Face & Gesture Recognition and Workshops*, Santa Barbara CA, USA, 2011.

[26] P. Ekman, "An Argument for Basic Emotions," *Cognition and Emotion* , pp. 196-200, 1992.

[27] K. Amaya, A. Bruderlin and T. Calvert, "Emotion from Motion," in *Graphics Interface*, Toronto, Canada, 1996.

[28] A. P. Atkinson, M. L. Tunstall and W. H. Dittrich, "Evidence for Distinct Contributions of Form and Motion Information to the Recognition of Emotions from Body Gestures," *Cognition*, vol. 104, no. 1, pp. 59-72, 2007.

[29] G. Castellano, S. D. Villalba and A. Camurri, "Recognizing Human Emotions from Body Movement and Gesture Dynamics," *Affective Computing and*

Intelligent Interaction, vol. 4738, pp. 71-82, 2007.

[30] O. Amft, H. Junker and G. Troster, "Detection of Eating and Drinking Arm Gestures Using Inertial Body-Worn Sensors," *Wearable Computers,* pp. 160-163, 18-21 October 2005.

[31] A. Pentland, "Looking at People: Sensing for Ubiquitous and Wearable Computing," *Pattern Analysis and Machine Intelligence,* vol. 22, no. 1, pp. 107-119, January 2000.

[32] T. Choudhury, S. Consolvo, B. Harrison, J. Hightover, A. LaMarca, L. LeGrand, A. Rahimi, A. Rea, G. Borrielo, B. Hemingway, P. Klasnja, K. Koscher, J. A. Landay, J. Lester and D. Wyatt, "The Mobile Sensing Platform: An Embedded System for Activity Recognition," *IEEE Pervasive Computing,* vol. 7, no. 2, 2008.

[33] D. Ververidis and C. Kotropoulos, "Fast and Accurate Sequential Floating Forward Feature Selection with the Bayes Classifier Applied to Speech Emotion Recognition," *Signal Processing*, vol. 88, no. 12, pp. 2956-2970, 2008.

[34] J. Nicholson, K. Takahashi and R. Nakatsu, "Emotion Recognition in Speech Using Neural Networks," in *Neural Information Processing*, Perth, Australia, 1999.

[35] B. Schuller, S. Reiter, R. Muller, M. Al-Hames, M. Lang and G. Rigoll, "Speaker Independent Speech Emotion Recognition by Ensemble Classification," in *Multimedia and Expo,* Amsterdam, The Netherlands, 2005.

[36] Y.-L. Lin and G. Wei, "Speech Emotion Recognition Based on HMM and SVM," in *Machine Learning and Cybernet*, Guangzhou, China, 2005.

[37] P. Rani, J. Sims, R. Brackin and N. Sarkar, "Online Stress Detection Using Psychophysiological Sygnals for Implicit Human-Robot Cooperation," *Robotica,* vol. 20, no. 06, pp. 673-685, 2002.

[38] J. Zhai and A. Barreto, "Stress Detection in Computer Users Based on Digital Signal Processing of Noninvasive Physiological Variables," in *Engineering of Medicine and Biology Society*, New York, USA, 2006.

[39] J. Zhai, A. B. Barreto, C. Chin and L. Chao, "Realization of Stress Detection Using Psychophysiological Signals for Improvement of Human-Computer

Interaction," in *SoutheastCon*, 2005.

[40] H. Gunes and M. Piccardi , "Fusing Face and Body Display for Bi-modal Emotion Recognition: Single Frame Analysis and Multi-frame Post Integration," *Active Computing and Intelligent Interaction,* vol. 3784, pp. 102-111, 2005.

[41] J. Wagner, E. Andre and F. Jung, "Smart Sensor Integration: A Framework for Multi-Modal Emotion Recognition in Real-Time," in *Active Computing and Intelligent Interaction and Workshops*, Amsterdam, The Netherlands, 2009.

[42] L. Kessous, G. Castellano and G. Caridakis, "Multimodal Emotion Recognition in Speech-Based Interaction Using Facial Expressions, Body Gesture and Acoustic Analysis," *Journal of Multimodal User Interfaces,* vol. 3, no. 1-2, pp. 33-48, 2010.

[43] G. Caridakis, G. Castellano, L. Kessous, A. Raouzaiou, L. Malatesta, S. Asteriadis and K. Karpouzis, "Multimodal Emotion Recognition from Expressive Faces, Body Gestures and Speech," in *Artificial Intelligence Applications and Innovations*, Athens, Greece, 2007.

[44] P. Dutta, P. M. Aoki, N. Kumar, A. Mainwaring, C. Myers, W. Willett and A. Woodruff, "Common Sense: Participatory Urban Sensing Using a Network of Handheld Air Quality Monitors," in *SenSys*, New York, USA, 2009.

[45] P. M. Aoki, R. J. Honicky, A. Mainwaring, C. Myers, E. Paulos, S. Subramanian and A. Woodruff, "A Vehicle For Research: Using Street Sweepers to Explore the Landscape of Environmental Community Action," in *CHI*, New York, USA, 2009.

[46] V. Padmanabhan and R. Ramjee, "Nericell: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones," in *ACM Sensys*, North Carolina, USA, 2008.

[47] M.-R. Ra, B. Liu, T. F. La Porta and R. Govindan, "Medusa: A Programming Framework for Crowd-Sensing Applications," in *MobiSys* , New York, USA, 2012.

[48] D. N. Lane, H. Lu, D. Peebles, D. Choudhury and A. T. Campbell, "A Survey of Mobile Phone Sensing," *IEEE Communication Magazine,* 2010.

[49] N. D. Lane, S. B. Eisenman, M. Musolesi, E. Miluzzo and A. T. Campbell, "Urban Sensing Systems: Opportunistic or Participatory," in *HotMobile*, NY USA, 2008.

[50] R. K. Ganti, F. Ye and H. Lei, "Mobile Crowdsensing: Current State and Future Challenges," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32-39, 2011.

[51] K. K. Rachuri, M. Musolesi, C. Mascolo, P. J. Rentfrow, C. Longworth and A. Aucinas, "EmotionSense: A Mobile Phone Based Adaptive Platform for Experimental Social Psychology Research," in *UbiConp*, Copenhagen Denmark, 2010.

[52] S. B. Eisenman, E. Miluzzo, D. N. Lane, R. A. Peterson, G. S. Ahn and A. T. Campbell, "The BikeNet Mobile Sensing System for Cuclist Experinece Mapping," in *SenSyn*, Sydney Australia, 2007.

[53] K. Lorincz, B.-r. Chen, G. W. Challen, A. R. Chowdhury, S. Patel, P. Bonato and M. Welsh, "Mercury: A Wearable Sensor Network Platform for High-Fidelity Motion Analysis," in *SenSyn*, California USA, 2009.

[54] N. Aharony, W. Pan, C. Ip, I. Khayal and A. Pentland, "Social fMRI, Investigating and Shaping Social Mechanism in the Real World.," *Pervasive and Mobile Computing*, vol. 7, pp. 643-659, 2011.

[55] J. Kukkonen, E. Lagerspetz, P. Nurmi and M. Andersson, "BeTelGeuse: A Platform for Gathering and Processing Situational Data," *IEEE Pervasive Computing*, vol. 9, 2009.

[56] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin and N. Triandopoulos, "AnonySense: Privacy-Aware People-Centric Sensing.," in *MobiSys*, Colorado USA, 2008.

[57] S. Dernbach, B. Das, N. C. Krishnan, B. L. Thomas and D. J. Cook, "Simple and Complex Activity Recognition through Smart Phones," in *Intelligent Environments*, Guanajuato, Mexico, 2012.

[58] S. P. Hall and E. Anderson, "Operating Systems for Mobile Computing," *Journal of Computing Sciences in Colleges*, vol. 25, no. 2, pp. 64-71, 2009.

[59] J. Rivera and R. van der Meulen, "http://www.gartner.com/," Gartner, 14 May

2013. [Online]. Available: http://www.gartner.com/newsroom/id/2482816.

[60] J. Ong, "TNW," 31 July 2014. [Online]. Available: http://thenextweb.com/google/2014/07/31/android-reached-record-85-smartphone-market-share-q2-2014-report/.

[61] R. Hitchens, "Selectors," in *Java NIO*, O'REILLY, 2002, pp. 117-144.

[62] Google, "Google-Gson," Google Inc., May 2014. [Online]. Available: https://code.google.com/p/google-gson/.

[63] N. Aharony, W. Pan, C. Ip, I. Khayal and A. Pentland, "FUNF," MIT, 2013. [Online]. Available: http://inabox.funf.org/. [Accessed 2014].

[64] "Google Play," MIT, 18 April 2013. [Online]. Available: https://play.google.com/store/apps/details?id=edu.mit.media.funf.journal&hl=en. [Accessed May 2014].

[65] P. Wu, J. Zhu and J. Y. Zhang, "MobiSense: A Versatile Mobile Sensing Platform for Real-World Applications," in *Springer Science and Business Media*, New York, 2012.

[66] Y. Wang, J. Lin, M. Annavaran, Q. A. Jacobson, J. Hong, B. Krishnamachari and N. Sadeh, "A Framework on Energy Efficient Mobile Sensing for Automatic User State Recognition," in *MobiSys*, New York USA, 2009.

[67] D. Hasenfratz, O. Saukh, S. Sturzenegger and L. Thiele, "Participatory Air Pollution Monitoring Using Smartphones," in *2nd International Workshop on Mobile Sensing* , Bejing China, 2012.

[68] A. Hunt and D. Thomas, The Pragmatic Programmer: From Journeyman to Master, Massachusetts, USA: Addison Wesley Longman, Inc., 2010.

[69] P. Tarr, H. Osscher, W. Harrison and S. M. Sutton, "N Degrees of Separation: Multi-Dimensional Separation of Concerns," in *21st International Conference on Software Engineering*, New York USA, 1999.

[70] Y. Shi and M. Larson, "First Approaches to Automatic Boredom Detection: DMIR Tackles the MediaEval 2010 Affect Task," in *MediaEval*, Pisa, Italy, 2010.

[71] H. Lee, Y. S. Choi, S. Lee and P. I.P., "Towards Unobtrusive Emotion Recognition for Affective Social Communication," in *Consumer Communications and Networking Conference*, Las Vegas, USA, 2012.

[72] G. Castellano, S. D. Villalba and A. Camurri, "Recognizing Human Emotions from Body Movement and Gesture Dynamics," in *Second International Conference ACII*, Lisbon, Portugal, 2007.