Energy efficient distributed wearable physiological monitoring: framework and implementations

Mladen Milošević

Follow this and additional works at: https://louis.uah.edu/uah-dissertations

Recommended Citation
Milošević, Mladen, "Energy efficient distributed wearable physiological monitoring: framework and implementations" (2013). Dissertations. 16.
https://louis.uah.edu/uah-dissertations/16

This Dissertation is brought to you for free and open access by the UAH Electronic Theses and Dissertations at LOUIS. It has been accepted for inclusion in Dissertations by an authorized administrator of LOUIS.
ENERGY-EFFICIENT DISTRIBUTED WEARABLE PHYSIOLOGICAL MONITORING: FRAMEWORK AND IMPLEMENTATIONS

by

MLADEN MILOŠEVIĆ

A DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in The Department of Electrical and Computer Engineering to The School of Graduate Studies of The University of Alabama in Huntsville

HUNTSVILLE, ALABAMA

2013
In presenting this thesis in partial fulfillment of the requirements for a doctor of philosophy degree from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department or the Dean of the School of Graduate Studies. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

MLADEN MILOSEVIC

MLADEN MILOSEVIĆ (date)
Submitted by Mladen Milošević in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Engineering and accepted on behalf of the Faculty of the School of Graduate Studies by the thesis committee.

We, the undersigned members of the Graduate Faculty of The University of Alabama in Huntsville, certify that we have advised and/or supervised the candidate on the work described in this dissertation. We further certify that we have reviewed the dissertation manuscript and approve it in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Engineering.

Committee Chair

Committee Member

Committee Member

Committee Member

Committee Member

Committee Member

Department Chair

College Dean

Graduate Dean
ABSTRACT

School of Graduate Studies
The University of Alabama in Huntsville

Degree Doctor of Philosophy

College/Dept. Electrical and Computer Engineering

Name of Candidate Mladen Milošević

Title Energy-Efficient Distributed Wearable Physiological Monitoring: Framework and Implementations

Recent technological advances in sensors, wireless networking, mobile and cloud computing promise to fundamentally change the way health care services are delivered and used. The development and proliferation of the wearable physiological monitors enable a shift in healthcare services from centralized hospitals and medical centers to individuals and their homes. Continuous wearable health monitoring has the potential to engage users, improve wellness management, prevent disease by early detection, and assist rehabilitation and treatments.

This dissertation presents a general framework and a practical implementation of computing infrastructure (both hardware and software aspects) to support mobile health and wellness monitoring applications as well as services to support further research in the area of wearable health monitoring. Specifically, we focus on a multi-tiered organization of the mHealth infrastructure that includes wearable sensor nodes at Tier 1, personal applications running on mobile computing platforms at Tier 2, and servers and services at Tier 3. We consider several important aspects of the mHealth systems including time-synchronization to support simultaneous distributed real-time monitoring and techniques for quantifying and improving energy-efficiency. These aspects are discussed on each tier of our infrastructure. Finally, the dissertation makes the case and introduces several original mHealth applications that are fully implemented and tested. These include monitoring and capturing dynamic heart response to posture transitions,
quantification of timed-up-and-go tests, monitoring of physical activity of wheelchair users, and monitoring of occupational stress of nurses.

Abstract Approval: Committee Chair

(Dr. Emil Jovanov) (date)

Department Chair

(Dr. Robert Lindquist) (date)

Graduate Dean

(Dr. David Berkowitz) (date)
ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Dr. Emil Jovanov, for his excellent guidance, caring, and patience from the very first day of my graduate studies. He was an excellent advisor in my professional and private life, and his great expertise helped me to acquire many new skills.

I would like to thank Dr. Aleksandar Milenković for his patience, motivation, enthusiasm, and immense knowledge. His support and guidance during my research were irreplaceable.

Also, I would like to thank Dr. Kerstin Thurow for all help and advice she provided. I would like to thank Dr. Karen Frith for her help during our collaborative projects. I would like to thank Tommy Shrove and Armen Dzhagaryan for their help during my research.

This work was supported in part by the U.S. National Science Foundation under grant CNS-1205439.

Finally, I would like to thank my beloved spouse Marina. She was always there cheering me up and stood by me through the good times and bad.
Table of Contents

List of Tables ................................................................................................................................................. xi

List of Figures .................................................................................................................................................. xii

Chapter One

Introduction .......................................................................................................................................................... 1

1.1 Background and Motivation .................................................................................................................. 1
1.2 Technological Trends ............................................................................................................................. 2
1.3 Wearable Physiological Monitoring .................................................................................................... 3
1.4 Main Contributions .................................................................................................................................. 5
1.5 Thesis Outline ......................................................................................................................................... 6

Chapter Two

Wearable Physiological Monitoring Infrastructure ...................................................................................... 7

2.1 System Architecture .............................................................................................................................. 7
2.2 mHealth System Architecture ............................................................................................................ 9
2.3 Tier 1 of mHealth Infrastructure ......................................................................................................... 11
2.3.1 Software Architecture .................................................................................................................... 11
2.3.2 Commercially Available Sensors .................................................................................................. 13
2.3.3 Custom Sensors from UAHuntsville ............................................................................................. 15
2.3.4 Built-in Smartphone’s Sensors ....................................................................................................... 16
2.4 Tier 2 of mHealth Infrastructure ......................................................................................................... 17
2.4.1 Personal Application Architecture ................................................................................................ 17
2.4.2 PC Applications ................................................................. 19
2.4.3 Smartphone Applications .................................................. 24
2.5 Tier 3 of mHealth Infrastructure .......................................... 28
   2.5.1 Software Architecture .................................................. 28
   2.5.2 mHealth Server Applications ........................................ 30

Chapter Three
Distributed Physiological Monitoring in mHealth Systems .................. 36
   3.1 Distributed Systems and Monitoring .................................... 36
      3.1.1 Time Measurement .................................................. 37
      3.1.2 Time Synchronization ............................................... 39
   3.2 Distributed Physiological Monitoring on Tier 1 of the mHealth ...... 40
      3.2.1 Time Synchronization in WBAN ................................... 40
      3.2.2 Implementation of Distributed Wearable Monitoring on the Tier 1 42
   3.3 Distributed Physiological Monitoring on Tier 2 of the mHealth ...... 46
      3.3.1 Time Synchronization of Personal Devices ....................... 46
      3.3.2 Implementation of Distributed Wearable Monitoring on Tier 2 .... 50
   3.4 Distributed Physiological Monitoring on Tier 3 of the mHealth’ ....... 53
      3.4.1 Time Synchronization on Tier 3 .................................... 54
      3.4.2 Support for Distributed Implementation ........................... 54

Chapter Four
Power-efficient Wearable Monitoring ........................................... 56
   4.1 Environment for Automated Power Measurements .................... 56

viii
4.2  Power-Efficiency on Tier 1 of the mHealth ................................................................. 60

4.2.1  Power Optimization of Signal Processing ................................................................. 61

4.2.2  Power Optimization of Data Transfer ........................................................................ 69

4.3  Power-Efficiency on Tier 2 of the mHealth ................................................................. 71

4.3.1  Power Optimization of Data Transfer with and without Compressions ............... 73

4.4  Power-Efficiency on Tier 3 of the mHealth ................................................................. 77

Chapter Five

Implementations and Original mHealth Applications ......................................................... 80

5.1  Real-time Assessment of Physiological Response to Posture Transitions .............. 81

5.1.1  Related Work .............................................................................................................. 83

5.1.2  Methods .................................................................................................................... 83

5.1.3  Smartphone Application ........................................................................................... 87

5.1.4  Experimental Setup .................................................................................................. 88

5.1.5  Results, Discussion, and Conclusion ....................................................................... 89

5.2  Real-time Quantification of Timed-Up-and-Go Test ...................................................... 90

5.2.1  Related Work ............................................................................................................ 91

5.2.2  Methods .................................................................................................................... 92

5.2.3  Smartphone Application ........................................................................................... 98

5.2.4  Results, Discussion, and Conclusion ....................................................................... 99

5.3  Real-time Monitoring of Activity of Wheelchair Users .............................................. 100

5.3.1  Related Work ............................................................................................................ 101

5.3.2  Methods .................................................................................................................... 102
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3.3</td>
<td>Smartphone Application</td>
<td>110</td>
</tr>
<tr>
<td>5.3.4</td>
<td>Results, Discussion, and Conclusion</td>
<td>112</td>
</tr>
<tr>
<td>5.4</td>
<td>Real-time Monitoring of Occupational Stress of Nurses</td>
<td>113</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Related Work</td>
<td>115</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Methods and Experimental Setup</td>
<td>116</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Participants and Simulation protocol</td>
<td>117</td>
</tr>
<tr>
<td>5.4.4</td>
<td>Signal Processing</td>
<td>118</td>
</tr>
<tr>
<td>5.4.5</td>
<td>Results</td>
<td>118</td>
</tr>
<tr>
<td>5.4.6</td>
<td>Discussion and Conclusion</td>
<td>119</td>
</tr>
<tr>
<td>5.4.7</td>
<td>Future Work</td>
<td>120</td>
</tr>
</tbody>
</table>

Chapter Six

Conclusion .............................................................................................................. 121

References ............................................................................................................ 124
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Real-time synchronization performance</td>
<td>45</td>
</tr>
<tr>
<td>2 Summary of possible time synchronization algorithms and their characteristic on smartphone</td>
<td>50</td>
</tr>
<tr>
<td>3 Evaluated Hardware and Software Configurations</td>
<td>65</td>
</tr>
<tr>
<td>4 Execution time, code size, and complexity for evaluated configurations</td>
<td>68</td>
</tr>
<tr>
<td>5 Total power, air date rate, and energy/bit of the typical wireless technologies</td>
<td>70</td>
</tr>
<tr>
<td>6 Critical parameters in the system design of the WBAN communication system</td>
<td>71</td>
</tr>
<tr>
<td>7 Lossless Compression Utilities</td>
<td>75</td>
</tr>
<tr>
<td>8 Benchmarks and their application domains used for energy efficiency estimation</td>
<td>78</td>
</tr>
<tr>
<td>9 The average and the standard deviation of a subset of parameters characterizing dynamic heart response to sit-to-stand transitions for 7 subjects (S1-S7)</td>
<td>90</td>
</tr>
<tr>
<td>10 Parameters for iTUG Characterization</td>
<td>95</td>
</tr>
<tr>
<td>11 iTUG Parameters for 3 Individuals with diagnosed Parkinson’s disease (S1-S3) and 4 healthy individuals (S4-S7)</td>
<td>100</td>
</tr>
<tr>
<td>12 Summary of the selected physiological parameters for all subjects</td>
<td>118</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>18</td>
</tr>
<tr>
<td>8</td>
<td>20</td>
</tr>
<tr>
<td>9</td>
<td>21</td>
</tr>
<tr>
<td>10</td>
<td>21</td>
</tr>
<tr>
<td>11</td>
<td>22</td>
</tr>
<tr>
<td>12</td>
<td>23</td>
</tr>
<tr>
<td>13</td>
<td>24</td>
</tr>
<tr>
<td>14</td>
<td>25</td>
</tr>
</tbody>
</table>
15 imWell HxM Android Application for interfacing Zephyr HxM sensors .......................................................... 26
16 Application architecture of the remote mHealth server ......................................................................................... 29
17 Session’s Data tables in mHealth Database ........................................................................................................... 31
18 Session’s Context tables in mHealth Database ....................................................................................................... 32
19 Event Annotations tables in mHealth Database ..................................................................................................... 33
20 User Management tables in mHealth Database ...................................................................................................... 34
21 Illustration of the Web API requests ..................................................................................................................... 35
22 Web Portal data visualization .................................................................................................................................... 35
23 Local and reference clock ticks .............................................................................................................................. 37
24 Clock drift of an ideal reference clock, and two non-ideal clocks ........................................................................ 38
25 Classification of time synchronization algorithms ................................................................................................ 39
26 System organization of the multi user system Avatar+. Time division multiplexing of synchronized controllers. ........................................................................................................................................ 43
27 Time division multiplexing for two users at maximum frequency (50Hz); IR LED is active when the signal is low. ........................................................................................................................................ 45
28 System organization for simultaneous monitoring of multiple subjects using personal devices ........................................................................................................................................... 51
29 Validation setup for time synchronization on the tier 2 .......................................................................................... 51
30 Waveforms for X axis of acceleration for two Zephyr BioHarness physiological monitors during validation process ........................................................................................................................................ 52
31 Time synchronization precision during validation test for two personal devices; one device is utilizing 4G cellular network, while the other is using local WiFi network; local times of the personal devices are intentionally set to be significantly different. ........................................................................ 53
32 Structure of the Records table in the mHealth Database .......................................................................................... 54
33 Experimental setup for automated power measurements ........................................................................................ 57
34 mPowerProfile graphical user interface [81] ......................................................................................................... 59
35 Daubechies D4 transform used in ECG processing .................................................. 63
36 An example of the R-peak detection algorithm; green circles represent potential R-peaks and red circles represent detected R-peaks ................................................................. 64
37 TIE language description of fixed-point multiplier ...................................................... 67
38 Energy profiles for hardware/software configurations ................................................. 68
39 Design space of the WBAN communication system .................................................. 71
40 Pandaboard .............................................................................................................. 74
41 Wireless data transfer energy efficiency of Pandaboard for compression under different frequencies EE.C[MB/Joule]; dotted lines represent the efficiency of uncompressed transmissions EE.UUP. .................................................................................. 76
42 Wireless data transfer energy efficiency of Pandaboard for decompression under different frequencies EE.D [MB/Joule]; dotted lines represent the efficiency of uncompressed transmissions EE.UDW.................................................................................. 76
43 Energy efficiency improvement comparing to single treded execution on 3.4GHz .......... 79
45 Modeling change of cardiovascular response induced by transition from sitting to standing 82
46 An example of physiological response during transition from sitting to standing: a) real-time heart rate calculated from RR intervals and hip position during transition; b) angle and magnitude calculated from accelerometer signals and hip position during transition. ........ 85
47 Algorithm for posture transition recognition.............................................................. 86
48 imWell monitoring of the cardiovascular response ................................................... 88
49 iTUG test phases and smartphone instrumentation of the subject .............................. 93
50 Smartphone orientation ............................................................................................ 94
51 Accelerometer and gyroscope signals (x, y, and z components) recorded on a smartphone mounted on a person with Parkinson’s disease during an iTUG test................................. 96
52 Angular velocity and upper body trunk angle during the sit-to-stand (left) and stand-to-sit (right) transitions ........................................................................................................ 98
53 sTUG: Smartphone TUG Android application screen displaying the parameters of the TUG test ................................................................. 99

54 Smartphone instrumentation of a wheelchair .................................................................. 103

55 Smartphone recordings of the x, y, and z components of the magnetic field for different wheelchair speeds: (a) 1 mph; (b) 2 mph; and (c) 4 mph .......................................................... 104

56 From signals to information: (a) z component of the magnetic field; (b) processed z component; (c) calculated speed; and (d) proximity sensor data ........................................... 105

57 Tracking tilt (a) and incline (b) ...................................................................................... 109

58 mWheelness Android application screens ....................................................................... 111

59 Heart Rate, Speed, and Hand Propelling during a freewheeling experiment .................. 113

60 Real-Time Physiological Monitoring Lab ....................................................................... 116
Chapter One

Introduction

Recent technological advances in sensors, wireless networking, mobile and cloud computing promise to fundamentally change the way health care services are delivered and used. Whereas traditional health-care systems are centralized and focused on reacting to illness, new health care systems based on wearable physiological monitors promise to shift the focus on proactive wellness management and ubiquitous health monitoring.

Advances and wide acceptance of communication and information technologies enable the delivery of medical services at a distance and change clinical practice by implementing electronic medical records, healthcare information systems, and ubiquitous health monitoring. These new technologies facilitate the development of a new set of tools for health care providers that can be used for wellness management, prevention, early detection, and treatment of a number of medical conditions.

1.1 Background and Motivation

The existing health care system is in eminent crisis caused by economic, social, and demographic trends. The overall health care expenditures in the United States reached $2.5 trillion in 2009 [1] and continue to grow. It is projected that the total health care expenditures will reach 4.5 trillion by 2020 or almost 20% of the Gross Domestic Product (GDP), threatening the wellbeing of the entire economy. At the same time, almost 50 million Americans (16.3 % of population) did not have health insurance in 2010 [2]. The uncompensated cost of care is up to $73 billion a year with nearly 2 million uninsured Americans hospitalized annually [3]. The
demographic trends are indicating two significant phenomena, aging population due to increased life expectancy and Baby Boomers demographic peak. Life expectancy has significantly increased from 49 years in 1901 to 77.6 years in 2003 and it is projected to reach 82.6 years by 2050 [4]. According to the U.S. Census Bureau, the number of elderly over age 65 is expected to double from 39.6 million in 2009 (or 13 percent of the total population) to nearly 70 million in 2025 with retirement of Baby Boomers, and reach 88.5 million by 2050 (or 20 percent of the total population) [5]. This trend is global, and the worldwide population over age 65 is expected to almost triple from 545 million in 2011 to 1.55 billion in 2050. These statistics underscore the need for more scalable and more affordable health care solutions.

1.2 Technological Trends

Advances in sensors, wireless networking, mobile and cloud computing promise to fundamentally change the way health care services are delivered and used. New services can be used for prevention [6], early detection [7], and treatment of a number of medical conditions [8]. Furthermore, chronic conditions such as diabetes [9], [10] or heart failure [11] can be managed more efficiently and with increased user’s convenience.

Availability, affordability, and excellent performance made smartphones the dominant personal computing platform. According to a report from April 2013, 216.2 million of smartphones was shipped worldwide in the first quarter of 2013, which represent an increase of 41.6% compared to the same period in 2012. Wide acceptance of smartphones has resulted in a shift in delivery of many services, such as, news, communication, and digital contents. Similar effect is visible in healthcare delivery [12]. With the recent explosion of the number of smartphones, a number of health monitoring and wellness applications has exponentially increased. According to the latest report from the March 2013, more than 97,000 mHealth applications are listed on a variety of application stores [13]. On the other side, an increasing number of users actively monitor their own health and fitness status [14]. The availability of
affordable smartphones and wearable devices and their widespread use and consumer acceptance create new opportunities for users and healthcare professionals.

1.3 Wearable Physiological Monitoring

Wearable physiological monitoring systems allow health and activity monitoring of users during their activities of daily living. Individuals can monitor changes in their physiological parameters and adjust behaviors in response to measurements [15], [16]. Integration of wearable systems with information systems used by health service providers facilitates automatic archiving of records in the personal health record and implementation of automated monitoring systems that can alert medical professionals if immediate interventions are needed [9], [10]. They can also be used for health monitoring of patients in ambulatory settings as a part of a diagnostic procedure or for supervised recovery from an acute event or surgical procedure [17]. An example system architecture for the wearable physiological monitoring systems is shown on Figure 1. A set of on or in-body sensors monitor a person’s physical activity and physiological signals. Sensors are controlled by and communicate with the personal server that integrates information from individual sensors and communicate with the medical server [18], [19]. The system may also integrate additional information, such as environmental conditions (e.g., local temperature and humidity) and provide access to authorized caregivers and other interested parties. Initial concept of the personal area networks for ubiquitous health monitoring has been proposed by Jovanov et al. and implemented at the University of Alabama in Huntsville in 2000 [20].
A number of research efforts have focused on wearable systems for health and wellness monitoring in the last decade. Researchers at the MIT Media Lab developed a research platform called MIThril [21], [22] - a wearable computing platform that combines body-worn sensing, computation, and networking enabling new applications in health, communications, and information delivery. A range of wireless medical sensors capable of collecting heart rate (HR), oxygen saturation (SpO2), electrocardiogram (ECG), and other physiological data was developed at Harvard Sensor Network Lab and named CodeBlue [23]. The first low-power WBAN for cardiac rehabilitation and activity monitoring has been implemented as collaborative effort between University of Alabama in Huntsville and Mayo Clinic in 2005 [17].

Most of the wearable physiological monitoring systems developed in research environment did not evaluate general framework for distributed wearable physiological monitoring. Moreover, many important design challenges such as energy-efficiency, time
synchronization, and efficient management of physiological records are not sufficiently addressed in the open literature. Most commercially available systems offering some kind of wearable physiological monitoring are focused on the fitness and wellness [24], [25], [26], [27], [28], with limited support for features important for researchers and health care professionals, such as precise time-stamping, event annotation, and raw signal archiving.

Aim of this dissertation is to develop a general framework and design space for energy-efficient distributed wearable physiological monitoring, address important design challenges, and provide several practical implementations as use cases. In the proposed and implemented 3-tiered architecture described in Figure 1, special focus is placed on energy-efficiency on each tier, efficient and flexible physiological records archiving, and precise time synchronization in distributed setup. In addition, a set of tools was developed to support system evaluation, and facilitate data collection and processing to enable further research in wearable physiological monitoring and its applications.

1.4 Main Contributions

Main contributions of the dissertation are as follows:

- The development of framework for distributed wearable physiological monitoring;
- An implementation of the mHealth infrastructure at the University of Alabama in Huntsville;
- An implementation of several desktop and smartphone applications utilizing mHealth infrastructure;
- An evaluation of the design space of wearable monitoring applications with recommendations for improved energy efficiency,
- An implementation of a low-power application-specific processor architecture; and
- The design, implementation, and evaluation of a synchronized distributed health and wellness monitoring.
1.5 Dissertation Outline

The rest of this dissertation is organized as follows. Chapter one introduces design framework and system architecture of wearable physiological monitoring systems, possible deployment configurations, and practical implementations of the system as well as its components. Principles and challenges of distributed wearable monitoring are presented in Chapter two. Distributed implementation issues are discussed on each level of the system architecture. Chapter three discusses energy efficiency of components in the wearable physiological monitoring systems. Several solutions for improved energy efficiency are presented. Chapter four present four original mHealth applications of wearable monitoring we implemented. For each original application we discuss a specific problem, its background, related work, and implemented solutions.
Chapter Two

Wearable Physiological Monitoring Infrastructure

Traditional health-care systems are designed and implemented to respond to symptoms or to manage medical conditions. Advances and wide acceptance of communication and information technologies enabled the delivery of medical services at a distance, also known as telemedicine [29]. Information and communication technologies also introduced changes in clinical practices, such as electronic medical records and healthcare information systems. All these changes contributed to the emergence of a new model of healthcare called eHealth [30]. Furthermore, new developments in sensors, wearable computing, and communications enabled wearable physiological monitoring, which can provide clinicians and users with tools and environments to gather physiological data over extended periods of time. Continuous monitoring can enable healthcare providers to focus on disease prevention and early intervention. Moreover, it can allow more efficiently managing of chronic conditions such as diabetes or heart failure. This emerging concept is also known as mHealth and represents the evolution of eHealth systems focused on wireless and mobile configurations [18], [31]. The following sections will describe general architecture of the infrastructure for wearable physiological monitoring and deployment configurations, and present a survey of practical implementations.

2.1 System Architecture

Main goals of a wearable physiological monitoring are continuous monitoring of person’s physiology during everyday activity, processing of monitored parameters, providing feedback to the user, and delivery of relevant information to healthcare providers. Wearable physiological
monitoring may also include additional information, such as environmental conditions (e.g., local temperature and humidity), to facilitate a better interpretation of monitored parameters. Considering aforementioned goals, a general architecture and data flow of a wearable physiological monitoring system can be represented using Figure 1.

Depending on the target population and deployment environment, infrastructure for wearable physiological monitoring can be deployed in several different configurations. As an example, deployment configuration for wearable monitoring of elderly persons in nursing homes and deployment configuration for monitoring healthy individuals during their daily activity will be significantly different. Although a number of configurations can exist, the most typical configurations are shown on Figure 2.

In the first configuration, illustrated in Figure 2 a), users wear sensors that communicate with a gateway placed in their environment, which further transfer data to a medical server over the Internet. This configuration is appropriate in cases where users spend most of their time in the known environment with limited area, such as hospitals, nursing homes, office, or home.

![Figure 2 Possible deployment configurations for wearable physiological monitoring](image)

*Figure 2 Possible deployment configurations for wearable physiological monitoring*
In the configuration shown in Figure 2 b), data from individual sensors are collected and aggregated on a portable personal device, which periodically transfers data to the medical server. Wide range of devices can be used as the personal device, such as laptops, tablets, and smartphones. Smartphones became the first choice because of their increasing proliferation, ever-improving functionality and performance, and affordability. Moreover, popularity and availability of smartphones made this configuration very popular and widely adopted. We based our mHealth architecture on this configuration. Detailed description of the mHealth architecture will be presented in Section 2.2.

Whereas the first two configurations require intermediary between sensors and the medical server, in the third configuration sensors communicate directly to the medical server over the Internet, as shown in Figure 2 c). This configuration allows ubiquitous monitoring with seamless integration of multiple sensors. In case feedback to the user is necessary, it can be achieved using personal device that will receive data from the medical server rather than from the wearable sensors. Unfortunately, limited energy density of today’s batteries and significant cost of communication over cellular network limit applicability of this configuration, but with future technology advances this configuration can be very successful.

2.2 mHealth System Architecture

The mHealth infrastructure is designed as a 3-tiered architecture with wireless body area sensor networks and other physiological monitors at Tier 1, personal computing devices at Tier 2, and mHealth servers at Tier 3, as represented in Figure 3.
Tier 1 consists of one or more Body Area Networks (BANs) or Body Sensor Networks (BSN) optimized for a specific health monitoring application. Each network integrates one or more wearable and intelligent sensor nodes. These sensors can sense vital signs (e.g., electrocardiogram, electroencephalogram, breathing rate, and body temperature), body motion (e.g., forces, angles, acceleration, and speed), posture, and environmental conditions (e.g., humidity, atmospheric pressure, and ambient light). Individual sensor networks may rely on a network coordinator to configure the network nodes, manage the nodes, and retrieve the data from the nodes. The network coordinator then communicates through wired or wireless connection to a personal application running on a personal device.

Personal applications running on a personal device (e.g., Android and Apple/IOS smartphones, tablets, or personal computers) represent Tier 2 of the proposed architecture. Applications are designed to facilitate (a) interface and management of a variety of sensors in the sensor network; (b) data retrieval from individual sensors, data logging and analysis to extract health status information, and (c) user interface providing real-time feedback with health parameters and recommendations (e.g., guided rehabilitation or exercise). The collected health status information is periodically uploaded to the mHealth servers over the Internet.
A group of servers providing storage, access, visualization, and support for data mining of physiological records forms Tier 3 of the mHealth infrastructure. Physiological records are stored in a database specifically designed to support efficient storage of variety physiological records and record annotations. Each record must have information about the subject, equipment used to collect records, and conditions under which the data are recorded. Physiological records can be organized by the type of application, and each record is precisely time-stamped.

2.3 Tier 1 of mHealth Infrastructure

Physiological monitoring starts with sensing of one or more signals. Wearable devices can sense a person’s physiological activity (e.g., heart and brain electrical activity, galvanic skin response), physical activity (e.g., body motion, body posture, body orientation), and/or environmental conditions (e.g., location, humidity, ambient temperature). Although there is a variety of sensors each capable of sensing different signals, a general software architecture of a physiological sensor can be represented using several high-level tasks. The following section contains details about aforementioned tasks and their interactions.

2.3.1 Software Architecture

Operation of a physiological sensor from Tier 1 can be described using four high-level tasks: signal sensing and conditioning, processing and feature extraction, communication, and power management, as shown in Figure 4.
The signal sensing and conditioning represents a point of entry for data in wearable physiological monitoring. Since correctness of the information at the point of entry is crucial for the entire system, sampling and precise time stamping of the samples should be implemented at this level. Some sensors implement signal conditioning and artifact removal on the sensor itself that results in reduced communication bandwidth requirements and power savings.

Sensed signals are often initially processed at the source on wearable devices. The processing can improve quality of the signals using software filtering, extract relevant features of the sensed signals to reduce amount of information that needs to be communicated, and aggregate signals' samples to improve efficiency and reliability of sensor communication.

The communication task transfers sensed and processed signals to Tier 2 of the mHealth infrastructure. It can be achieved using wired or wireless data transfer. Although a wireless transfer usually consumes more energy than a wired transfer, it is the most common type of transfer as it improves wearability and user convenience. In order to ensure data privacy and security, the communicated information is often encrypted.

Although power management task does not have a direct influence on the data flow, it is very important, even crucial in some instances, for overall wearable monitor operation. In order to improve battery life of the sensor and overall user experience, power management interacts
with all three tasks ensuring that the sensor utilizes low-power operation mode whenever possible. Furthermore, since wearable sensors can utilize energy harvesting / scavenging as a primary or an additional power source, power management should adjust overall operation of the sensor to maximize utilization of available energy.

We use three types of physiological sensors in our mHealth infrastructure: (i) commercially available sensors, (ii) custom made sensors at UAHuntsville, and (iii) smartphone’s built-in sensors. Each type will be described in the following sections.

2.3.2 Commercially Available Sensors

Recent advances in sensors technology and mass production allowed many companies to start making and selling variety of physiological sensors at affordable price [32], [33], [34], [35], [36], [37], [38]. Sensing heart activity, physical activity, brain activity, and other physiological signals is now easily accessible to researchers and people interested in monitoring their own health and wellness. Figure 5 shows a subset of commercially available physiological sensors used in this dissertation. Available sensors differ in sampling frequency, accuracy, type of extracted and communicated information, wireless interfaces used for communication, battery life, and price.

Heart activity is one of the most monitored physiological signals because of the prominence of cardiovascular conditions. Consequently there exists dozens of commercially available sensors capable of sensing heart electrical activity. Most heart rate monitors are produced in forms of chest belts (Figure 5 a-d ), and range from inexpensive sensors (less than $100) intended for fitness tracking to more sophisticated heart monitors designed for research with price over $2,000. Whereas fitness level heart rate monitors can report only average heart rate calculated every few seconds [39], [40], sophisticated heart monitors [33], [35] can also report interbeat intervals (RR intervals) and electrocardiogram (ECG), and store all the information in their local memory. This feature is essential for the heart rate variability analysis
On the other side, fitness grade heart rate monitors have significantly longer battery life and smaller form factors than sophisticated monitors.

Figure 5 Commercially available physiological sensors used on Tier 1: a) Garmin ANT+ heart monitor, b) Zephyr HxM heart rate monitor c) Zephyr BioHarness 3 physiological monitor, d) Hidalgo Equivital 2 physiological monitor, e) Garmin ANT+ bike and cadence sensor, f) Garmin ANT+ foot pod, g) Zeo sleep monitor, h) NeuroSky MindSet EEG sensor, i) Emotiv EGS neuroroheadset

We use both type of sensors depending on requirements. In the case we need simple, smaller monitors with longer battery life, we use Garmin ANT+ heart rate monitors (Figure 5 a)) or Zephyr HxM heart rate monitor (Figure 5 b)). The Garmin monitor uses a low-power ANT+ wireless interface which enables it to work for several months on a single coin battery. Unfortunately, ANT+ wireless interface is not widely adopted and supported on smartphones, which limits its compatibility. Zephyr HxM heart rate monitor uses a widely adopted Bluetooth wireless interface, but it suffers from significantly shorter battery life. In the case we need a sophisticated heart monitor, we use Zephyr BioHarness 3 (see Figure 5 c)) and Hidalgo Equivital 2 (Figure 5 d)) physiological monitors. Both devices can record and communicate RR intervals and ECG signals. In addition, both monitors offer access to several additional sensors such as a 3-axis acceleration sensor and a respiration sensor. The Bluetooth wireless interface is used for communication on both devices.

Sensors for monitoring physical activity used in this dissertation include Garmin ANT+ foot pod sensors (Figure 5 e)) and Garmin ANT+ bike and cadence sensors. The foot pod sensor
measures the number of steps made and speed during walking/running, while the bike and cadence sensor measures cycling speed. Both sensors use the low-power ANT+ wireless interface for communication with Tier 2.

Third group of commercially available sensors we use are for monitoring brain electrical activity. We use Zeo sleep monitors (Figure 5 g), NeoroSky MindSet EEG sensors (Figure 5 h), and Emotiv EEG neuroheadsets (Figure 5 i). The Zeo sleep monitor is a low-power headband with a single channel EEG intended for sleep studies. The MindSet EEG provides a single channel EEG in the form of a wireless headset, whereas the Emotiv EEG headset offers 14 channels of EEG sampled at 2 KHz, filtered, and reported with the rate of 128 samples per second for each channel.

2.3.3 Custom Sensors from UAHuntsville

Physiological and physical activity sensors are increasingly commercially available; however, there is a need for custom sensors with features and characteristics that cannot be found on the market.

Typical example is Avatar, a system for real-time control of avatars in virtual training space, developed at UAH for Northrop Grumman. The system integrates optical, wired, and wireless sensors to achieve a cost-effective, unobtrusive, real-time body position monitoring [42]. The system uses absolute positioning of a few reference points using infrared diodes on user’s body, whereas the position and orientation of other body segments is determined using a network of inertial sensors embedded in the clothes. The inertial sensors are connected through a wired serial interface to a master node, iControl, that acts as the system gateway, as shown in Figure 6. iControl controls a network of inertial sensors and communicates wirelessly via Bluetooth with a workstation. Positioning of a reference point, usually the master node placed on the user’s shoulder, is achieved using two Wii remotes. The reference point position is determined by triangulation. The Wii remotes communicate with the workstation wirelessly using a Bluetooth
interface. The individual inertial sensors, iSense, are designed as intelligent 5 degree of freedom (5DOF) sensors that communicate with the gateway through the shared I²C bus. The system architecture and the sensors are shown on Figure 6. Recently our team developed a high performance 9 DOF inertial sensor with low power wireless interface for embedded monitoring applications [43].

![Figure 6 Avatar system for real-time body position monitoring. a) System Architecture b) iSense Sensor Node c) iControl Gateway Node](image)

### 2.3.4 Built-in Smartphone’s Sensors

In the past few years smartphones not only increased computing capabilities, but also became equipped with a number of sensors. A typical example is Google Nexus 4 smartphone [44]. It includes a three-axis accelerometer, a three-axis gyroscope, a three-axis magnetometer, a barometer, a proximity sensor, an ambient light sensor, a GPS (Global Positioning System), and two cameras. Nexus 4 supports several communication interfaces such as 3G/4G Cellular, WiFi, Bluetooth, and NFC (Near Field Communication).

Each of these sensors has been integrated with a specific role in mind. The accelerometer measures smartphone orientation using Earth’s gravity and is typically used to keep the screen
regardless of the smartphone orientation. The magnetic sensor is sensing the Earth’s magnetic field and is normally used as compass to aid navigation by determining smartphone’s orientation. The proximity sensor is typically used to determine when the smartphone is brought up to the user’s ear to prevent accidental commands and save power by turning off the screen. The ambient light sensor is used to automatically adjust screen brightness.

Since major mobile operating systems, such as Android, iOS, and Windows 8, support frameworks for managing the sensors, including continual sampling, embedded sensors can be used for wearable monitoring applications. Several implementations of original wearable monitoring applications using smartphone’s built-in sensors are presented in sections 5.2 and 5.3.

2.4 Tier 2 of mHealth Infrastructure

Physiological signals, sensed by sensors on Tier 1, are received and processed on Tier 2. Whereas most of the sensors we use are commercially available and can be used for a number of applications, each specific type of physiological monitoring requires application-specific signal processing and feature extraction. Due to the inherent variation of human physiology the most successful applications use personalization of processing and presentation [45]. Although personal applications can utilize a variety of sensors, signal processing procedures, and user interfaces, all applications can be described using several high-level tasks. Implementing each of the tasks in a separate module of personal health application can improve application organization reusability, and facilitate further application development. The following section describes software architecture of a generic personal health or wellness application, as well as its processing modules and their interactions.

2.4.1 Personal Application Architecture

Our design contains six main modules as follows: (a) sensor interface, (b) general signal processing, (c) application specific signals processing, (d) user interface, (e) local storage, and (f) remote database access module, as presented in Figure 7.
The sensor interface module handles all communication with the sensors nodes. Flexible design of this module is important since multiple sensor nodes with different characteristics on the Tier 1 can provide the same type of information. Using this approach, a change of sensor in Tier 1 requires only a change in the sensor interface module. Moreover, this module is the same for every application utilizing the same set of sensors, and consequently it can be entirely reused.

The signal processing module performs generic signal processing tasks. For example, gyroscope data stream might come from the chest belt or activity sensor on ankle, but typical filtering and integration tasks are the same in both cases. The results are processed in the application specific processing module since each application requires specific processing procedures.

The user interface module collects user input and provides feedback to user. Certain sub-tasks of the user interface can be reused. For example, user screens for login and configuration often can be used unchanged or just slightly modified.

The local storage module handles data archiving on the personal device. The most convenient and frequently used approach is local SQLite database [46] and CSV (Coma
Separated Values) files. In general, this module often needs only slight modification for data archiving in each new application.

The remote database module facilitates communication with the remote mHealth server. It provides data uploading to and fetching from the mHealth remote server. This module can usually be reused in each application without any modification.

We developed several personal applications for the wearable physiological monitoring utilizing the described modular design architecture. The applications are developed in Microsoft Windows or Google Android development environment and will be described in next two sections.

2.4.2 PC Applications

We developed several PC applications to support experimentation, sensor testing, and wearable monitoring applications. As examples, in this section we present mHealth ANT+ PC App, UAHuntsville Serial Port App, and mHealth Zephyr PC App.

**mHealth ANT+ PC App**

The mHealth ANT+ PC App [47] is designed to communicate with an ANT+ Garmin chest belt (Figure 5 a) and an ANT+ Garmin footpod (Figure 5 f). The application collects user’s heart activity recorded with the chest belt and physical activity recorded with the footpod, plots them in real-time, stores records locally, and uploads them to the remote mHealth server. The application allows loading and plotting of historical records from the local record collection and from the remote mHealth Database. Users can also annotate physiological records in real-time by entering their current type of activity or description of important application-specific events (e.g., chest pain during exercise). A time stamp of an event is automatically generated by the application, annotations are synchronized with the physiological records, and all records are uploaded to the remote mHealth server. A characteristic screenshot of the application is shown in Figure 8.
UAHuntsville Serial Port App

The UAHuntsville Serial Port App is developed to support development of custom communication protocols with sensors and other embedded devices. The application can be configured to receive data packets with a custom format, specifying the number of channels and data types (e.g., uint8, int16, uint32, float, double).

Figure 9 shows the configuration screen of the application. A user-friendly graphical interface allows user to set parameters of a custom protocol, such as packet header identification, packet length, number and type of samples, and type of the checksum used. Configuration of the each custom protocol can be stored in a configuration file and loaded when needed.

A screenshot illustrating real-time data plot is shown in the Figure 10. The application can plot up to six channels at a time. Furthermore, all received information can be stored into a CSV file. In addition to real-time plotting and recording, the application supports loading of pre-recorded data from a file and plotting them on a graph.
Figure 9 Configuration screen of the UAHuntsville Serial Port App

Figure 10 Real-time data plot screen of the UAHuntsville Serial Port App
mHealth Zephyr PC App

The mHealth Zephyr PC App [48] collects user’s heart activity (ECG, interbeat intervals, and heart rate), physical activity (three-axis acceleration), and breathing pattern (chest expansion and breathing rate) from a Zephyr BioHarness physiological monitor (Figure 5 c). Collected information can be plotted in real-time, stored locally, and uploaded to the remote mHealth server. Figure 11 shows real-time data plot in the application. The application is used to develop generic Sensor interface module for Zephyr BioHarness physiological sensor. Unique feature of the application is support for the experimentation which will be explained in the following section.

![Figure 11 mHealth Zephyr PC App for Zephyr BioHarness physiological monitor](image)

**System Support for Experimentation**

Research environment often requires structured experiments where subjects closely follow the predefined protocol. This is particularly important for certain medical experiments where all subjects must receive exactly the same instructions and follow the same steps during an
experiment. There are no programs on the market that support this functionality, and it was essential for our experiments and study of physiological responses induced by postural changes.

In order to facilitate structured experiments in controlled environments, we developed an application that can guide subjects through each phase of an experiment using audio and visual cues. A screenshot of the application used for user guidance is shown in Figure 12. A list of the phases of the experiment (activities) and their duration is located on the far left side of the screen. Middle panel shows current and right window presents next phase of the experiment, as well as remaining time to the start of the next phase of experiment. Before each experiment’s phase, application’s text to speech feature will also tell the user what is the next phase. Recorded physiological records will automatically be annotated at the beginning of each experiment phase using the name of the experimental phase.

![Integration of user guidance according to the experimental protocol and monitoring in the mHealth Zephyr PC App](image)

Recorded physiological data can be hard to understand, and sometimes it is not clear what subject was at the time data was recorded. This is particularly hard in the presence of artifacts. With this in mind, we enabled our application to synchronously receive and record data from wireless sensors and video from the web camera. During data review and algorithm development, the user can synchronously replay video and physiological signals, as shown in
Figure 13. Synchronous physiological data and video replay can significantly help in understanding cause of changes in physiological signals and the nature of artifacts. The system is also used to facilitate algorithm development.

2.4.3 Smartphone Applications

Our smartphone applications are developed for Google Android operating system, and follow described modular design. We present the following examples of our applications:

- **imWell** – wellness monitoring application using Zephyr BioHarness,
- **imWell HxM** – wellness monitoring application with support for Zephyr HxM,
- **imWell Posture** – wearable monitoring of physiological response to posture changes,
- **mWheelness** – smartphone based monitoring of activity of wheelchair users, and
- **sTUG** – smartphone based automated monitoring of Timed Up and Go procedure.

In the following sections each application will be briefly described. Implementation details, signal processing, and related work will be described in the Chapter 5 of the dissertation.

**imWell**

*imWell* (inconspicuous mobile wellness monitoring) is an Android application that interfaces the Zephyr BioHarness physiological monitor. The application receives user’s heart
activity, physical activity, and breathing pattern from the chest belt (Figure 5c). The application provides local storage of the physiological data in SQLite database and CSV files, and uploads data to the remote mHealth server. Furthermore, user can select what information will be stored locally and what will be uploaded to the remote server. User can also annotate physiological records in real-time with the current activity or important events. Screens of the imWell Android app are shown on Figure 14.

![Figure 14 imWell Android Application for Zephyr BioHarness physiological monitor](image)

The application is designed as a template application for the Zephyr BioHarness physiological monitoring and does not provide any specific signal processing. Applications imWell Posture and mWheelness are directly derived from this application.

**imWell HxM**

imWell HxM Android app interfaces the Zephyr HxM chest monitor. It receives user’s heart activity (interbeat intervals and heart rate) and physical activity (distance, cadence, and speed) (Figure 5b). Application provides local storage of the physiological data (SQLite database and CSV files), uploads data to remote server, and supports real-time data annotation. This application also allows user to select what information will be stored locally and what will be uploaded to the remote server. Screens of the imWell HxM Android app are shown in Figure 15.
Posture changes represent physical stress that challenges physiological homeostasis. It was hypothesized that cardiovascular response to posture changes can be used to assess the state of the cardiovascular system and overall wellness status during activities of daily living [49]. Therefore, individual responses represent simplified “stress tests” of our system. *imWell Posture* is developed to automatically detect transitions between postures, such as sitting to standing, and assess the dynamic physiological response to posture transitions during activities of daily living. We use Zephyr BioHarness 3 physiological monitor that continually reports heart activity and physical activity via Bluetooth to the application running on the smartphone. The application processes all the data in real-time, recognizes posture transitions from accelerometer data, characterizes dynamic heart response to posture changes, annotates, logs, and uploads heart activity data to our mHealth server as a personal health record of individual event. Individual responses are influenced by a number of other factors, such as psychological state, ambient conditions, and others. However, continuous monitoring allows collection of dozen events a day that effectively cancels out all the other factors, leaving our cardiovascular status as the main contributing factor of the response. Moreover, day-by-day monitoring during longer periods
enables detection of long-term trends indicating improvement or deterioration of the cardiac and
health status. Details about signal processing, typical screens, and type and form of stored and
uploaded data will be presented in 5.1.

**mWheelness**

Increase in physical activity level can help to lower the risk of inactivity-related diseases,
such as cardiovascular disease, insulin resistance, hyperglycemia, and type 2 diabetes. Physically
inactive individuals are almost twice as likely to develop coronary heart disease when compared
to those who exercise regularly. People with limited ambulatory skills who use wheelchairs for
mobility are especially at high-risk for all inactivity-related diseases. For example, it has been
reported that a person with a spinal cord injury (SCI) has a significantly greater risk of mortality
from coronary heart disease (225%) than an able-bodied person [50].

In order to provide an affordable, reliable, and easy to use solution for monitoring the
physical activity of users who rely on wheelchairs for mobility we developed mWheelness
Android app, which utilizes a smartphone’s built-in sensors to capture and record physical
activity of manual wheelchair users in both unstructured and structured environments. It can also
collect user’s heart activity, physical activity, and breathing pattern from the Zephyr BioHarness
physiological monitor. Details about signal processing and typical screens will be presented
in 5.3.

**sTUG**

The Timed-Up-and-Go (TUG) is a frequently used clinical test for assessing balance,
mobility, and fall risk in elderly population and people with Parkinson's disease [51]. It is simple
and easy to administer in an office, and thus can be used in screening protocols. The test
measures the time a person takes to rise from a chair, walk three meters, turn around, walk back
to the chair, and sit down. Longer TUG times have been associated with mobility impairments
and increased fall risks [51], [52], [53].
We developed a smartphone application called sTUG that completely automates the iTUG test so it can be performed at home. sTUG captures the subject’s movements utilizing smartphone’s built-in accelerometer and gyroscope sensors, determines the beginning and the end of the test and quantifies its individual phases. All measures can be automatically uploaded into a personal health record on a medical server. Details about TUG test, signal processing, typical screens, and test descriptors will be presented in 5.2.

2.5 Tier 3 of mHealth Infrastructure

Tier 3 of the mHealth infrastructure integrates data from the Tier 2 and provides permanent physiological data storage (personal health record), easy access to data for authorized users, and support for data mining. Aforementioned functionalities also represent main tasks on the Tier 3 which can be used to describe its global software architecture.

2.5.1 Software Architecture

Our design of the Tier 3 of the mHealth infrastructure contains four modules: web API, data storage, data processing and mining, and data access and visualization, as presented in Figure 16.

The web API accepts physiological data from the personal devices and stores it into the provided data storage - database. It also allows personal devices access to data already stored in the database. Since the database contains private health information, web API must require authentication before it allows personal devices to store or retrieve information from the database. Moreover, communication between personal devices and web API should be encrypted to provide secure data transfer.
Figure 16 Application architecture of the remote mHealth server

Main purpose of the data storage module is to provide permanent data storage for physiological records. Typically the data storage module is implemented as traditional relational database, since it provides flexibility of data models, allows easy access to a subset of data using standardized queries, and utilize proven, stable, and reliable database management system. Relational model of the database should be carefully tailored to provide flexible and efficient physiological data storage.

The data access and visualization module should allow authorized users easy access to recorded data. Since the database typically contains physiological data from multiple users, the module must restrict access of each user to a subset of data he/she is authorized to access. Moreover, the module may implement adequate visualization to facilitate and accelerate visual inspection of the records. Visualization can be achieved using charts, tables and/or similar data representation methods.

All data processing that can be implemented in real time or near real time should be implemented on the Tiers 1 and 2, rather than on the Tier 3. Aforementioned approach will better distribute processing load, reduce communication, and improve scalability of the whole system. Processing on Tier 3 is implemented in data processing and mining module.
2.5.2 mHealth Server Applications

Our servers on the Tier 3 of the mHealth infrastructure are running a free operating system Ubuntu Server, and they are designed and implemented to work as virtual machine appliances in either open source VM VirtualBox [54] or proprietary VMWare [55] environment. This approach follows server virtualization trend and offers several advantages. The main advantages of the aforementioned approach are easy deployment and migration to new physical platforms. Furthermore, many cloud computing providers, such as Amazon [56], support importing and exporting of the virtual machine appliances to their infrastructure.

The Tier 3 of the mHealth infrastructure is composed of three main components: mHealth Database, Web API, and Web Portal. mHealth Database is developed as open source Oracle’s MySQL [57] relational database, and utilizes procedures for data insertion and access. Web API is developed in PHP [58], prevents unauthorized insertion and retrieval of data, and utilizes https protocol to provide secure data communication. Web Portal enables authorized users, such as physician and authorized caregivers, to access physiological records. Authorized users can access only a subset of records they are authorized to access. It is developed in Sencha JS [59] framework.

mHealth Database

We designed our database to provide not only storage for physiological record and context of their recording, but also support for management and guidance of a variety of experiments in research environment. Experiments can be conducted using a specific protocol, have authorized investigators, a list of sessions with individual participants, and individual physiological, activity, and multimedia records.

Current implementation of mHealth Database contains 23 tables. All the tables can be divided into four categories based on their functions. The categories are: Session’s Data, Session’s Context, Event Annotations, and User Management.
Session’s Data Tables

The tables in Session’s Data category are shown on Figure 17. These three tables represent core of the database and contain actual physiological records and references to their context. For every session recorded new row is added into the table session. Session table contains references to a subject of the recording, a personal device used for data uploading, an experiment it belongs, and a protocol used during recording if applicable. Additional information such as time of the beginning and end of recording are also stored in the session table.

During each session multiple sensors from Tier 1 can be used to record multiple signals. Therefore, multiple physiological signals can be instanced into the session_signals table. For each signal we store type of signal, placement, type, and sampling frequency of the sensor used for recording, and format of stored data.

Each session signal may contain multiple instances in the records table. The records table contains actual physiological data with the precise time of the recording for the particular data set. There are two timestamps, one referring to local time in the place the data was recorded, and the other referring to the UTC time. The reason for this approach is to preserve information about ambient conditions (e.g., part of the day the data was recorded), as well as provide easy data synchronization across multiple time zones.

Figure 17 Session’s Data tables in mHealth Database
Session’s Context Tables

Tables sensor_types, sensors, positions, signals, experiments, investigators, servers, protocol, and protocol_items represent the Session Context group of tables, as shown in Figure 18.

The first four tables are populated in design time, and contain predefined values. Their modification is not performed during data recording. Table experiments can be accessed and modified only by administrator or principal investigator of the experiment to protect privacy of the participants. It contains information about experiment associated with multiple sessions of the physiological monitoring. Table investigators contains references to users that serve as investigators in particular experiments. A user defined as investigator will be able to access anonymized data using mHealth Web Portal. Table servers contains information about the personal device (personal sever) used for physiological data uploading.

If a specific protocol was used during session recording, tables protocol and protocol_items will contain information about the protocol. Each row in the protocol table represents one specific protocol, and it can instance multiple rows in the protocol_items table, which represent phases of the given protocol. Our previously described program provides system support for experimentation and utilizes this table to load specific protocol that will be used for guidance of participants.
Event Annotations

Annotations of significant events are very important part of experimentation and provide crucial information for the understanding of recorded data (e.g., chest pain that might be associated with ischemia during exercise). Support for management of annotations is implemented in the Event Annotations group of tables. This group of tables contains event_type, event_type_items, session_events, and session_event_items tables, shown in Figure 19. The first two tables contain predefined values and usually are populated in design time. Tables session_events and session_event_items contain actual physiological record annotations. The session_event_items table is used in case of complex annotations with multiple values. Example of this kind of annotations will be described in application for assessment of physiological response to posture transitions described in Chapter 5.

Figure 19 Event Annotations tables in mHealth Database

User Management

Experiments with human subjects require sophisticated management of users and privileges to protect privacy of participants [60]. We designed our User Management group of tables to support user management complied with minimum HIPAA requirements. Tables subjects, user, group, user_group, permission, user_permission, group_permission make User
Management group of tables, shown in Figure 20. The *subjects* table contains information about subjects of the recording, while the *user* table contains information about the user of the mHealth system. Among the other information, the *user* table stores username and password necessary for data uploading and access for each user. Tables *group*, *user_group*, *permission*, *user_permission*, and *group_permission* contain information about permissions in the system, their grouping, and permissions each user is granted.

![User Management tables in mHealth Database](image)

**Figure 20 User Management tables in mHealth Database**

**Web API**

Web API is designed to be intermediary between the personal devices and mHealth Database. It accepts data from the personal devices and stores it into the database, and also allows personal devices to retrieve stored data from the database. Any action using Web API requires the successful authentication. Authentication is implemented as the separated request, rather than as part of the each request. Upon successful authentication web session is created, allowing further execution of Web API requests without additional authentication. After predefined period of inactivity, web session automatically expires and the authentication process has to be repeated. Three characteristic Web API requests for authentication, data upload, and data retrieval are shown in Figure 21.

34
https://IPADDR/admin.php?action=authentication&username=USERNAME&password=PASSWORD
https://IPADDR/admin.php?action=Add_RR&ID_Session=IDSESSION&UTCtime=UTC&Text_data=DATA

Figure 21 Illustration of the Web API requests

In order to provide secure communication, Web API utilize HTTPS protocol [61] based on 256-bit AES CBC encryption (Advanced Encryption Standard with Cipher Block Chaining) [62].

Web Portal

Web Portal provides easy access to physiological data and its basic visualization. It requires only browser to access recorded session in mHealth Database. Each authenticated user is allowed to access only subset of a data he/she is authorized to access. The user can easily visualize data by selecting desired session and particular signal inside the session. Typical screen of the Web Portal application is shown in Figure 22.

Figure 22 Web Portal data visualization
Chapter Three

Distributed Physiological Monitoring in mHealth Systems

Computers are following a general trend of becoming more and more distributed and deeply embedded into the environment. Wearable physiological monitoring is no exception, and it can be implemented in distributed fashion on each tier of the proposed mHealth architecture. One or more subjects can be simultaneously monitored using multiple sensors that form a distributed Wireless Body Area Networks on Tier 1, multiple subjects can be simultaneously monitored using multiple personal devices on Tier 2, and/or databases for the physiological data storage can be implemented in distributed fashion on Tier 3. Each of these cases will be discussed separately. Before we describe possible and implemented support for distributed physiological monitoring we will present general principles and challenges in organization of distributed systems and their measurements.

3.1 Distributed Systems and Monitoring

Distributed system represents a “collection of independent computers that appear to the users of the system as a single computer” [63]. Factors such as decrease of computer price and size significantly contributed to appearance of distributed systems and there are several important benefits in building them, such as better price / performance ratio, increased reliability, easy incremental growth (scalability), and improved mobility. On the other side, building distributed systems also brings multiple challenges, some of which are designing, implementing, and using distributed software and creating and maintaining network between members of the system.
Since distributed system has to appear to the user as a single entity, it is important that all members of the system can make uniform observations. Moreover, uniform observation of time in case of distributed monitoring is necessary. Challenges for ensuring uniform observation of time in a distributed system and possible solutions will be discussed in the following sections.

### 3.1.1 Time Measurement

In order to make a consistent picture of the world, each element of the distributed system has to maintain information about time. Digital physical clock is a device for measuring time, which contains a counter and a physical oscillation mechanism that periodically generates an event, *microtick*, to increment the counter. Granularity of the digital clock is a duration between two consecutive microticks \((g=1/f)\) as shown in Figure 23. Finite size of the granularity leads to a digitalization error in time measurement.

*Reference clock* represents an external observer that can generate precise time-stamps for all relevant events in the system (Figure 23). For the reference clock we assume its counter is always the same as that of the international time standard, and we can disregard its digitalization error.

![Figure 23 Local and reference clock ticks](image)

The accuracy of the distributed measurement is determined by a very important characteristic of a digital clock – *clock drift*. The drift of a physical clock between its *microtick* \(i\) and *microtick* \(i+1\) is the ratio of periods between local clock and the reference clock, at the instant of *microtick* \(i\). Each microtick should contain \(n\) reference clock ticks. The drift is determined by measuring the duration of a microtick of the clock with the reference clock and dividing it by the nominal number of reference clock ticks in a clock’s microtick \((n)\) [64]:

\[
\text{reference clock’s tick: } \frac{z(\text{microtick}_i)}{z(\text{microtick}_{i+1})}
\]
Drift is influenced by the tolerances of the crystal oscillator, environmental conditions (e.g., change in the temperature), change in the voltage level that is applied to a crystal oscillator, and aging of the crystal. Within specified environmental parameters, the drift rate of an oscillator is bounded by the maximum drift rate $\rho_{\text{max}}$, which is documented in the data sheet of the resonator. Typical maximum drift rates are in the range of $10^{-2}$ to $10^{-7}$ s/s (seconds per seconds) or better, depending on the quality (and price) of the oscillator. Unit used to determine clock drift is typically $\text{ppm}$ - parts per million. It means that oscillator with frequency of 32MHz and accuracy of $\pm200\text{ppm}$, has frequency drift of:

$$\Delta f(\text{Hz}) = \frac{\pm200\text{ppm} \times 32\text{MHz}}{1000000} = \pm 6.4\text{kHz}$$

Every clock has a non-zero drift rate (change of drift), which means that free-running clocks (clocks that are never resynchronized) will leave any bounded relative time interval after a finite time, even if they are fully synchronized at startup. Figure 24 illustrates clock drifts in time for reference (ideal) clock, and for two non-ideal clocks. The difference between the times on two clocks is defined as skew.

![Figure 24 Clock drifts of an ideal reference clock, and two non-ideal clocks](image_url)
3.1.2 Time Synchronization

Multiple unrelated processes of a distributed system running on different devices should be able to make consistent decisions about the ordering of events in the system. In order to enable distributed system to make such decision, and having in mind existence of a clock skew and a clock drift as described in the previous section, we need periodical adjustments of each node’s time. This adjustment is called *time synchronization*.

Assuming that we have distributed system with set of nodes, each one with its own local physical clock $c$ that ticks with granularity $g$, all of the clocks are internally synchronized with a precision $\Pi$ if for any two clocks $j, k$ [64]:

$$|z(microtick^j_i) - z(microtick^k_i)| < \Pi \quad (3-3)$$

There are many proposed and implemented algorithms for time synchronization. Figure 25 shows a classification of time synchronization algorithms [65].

![Figure 25 Classification of time synchronization algorithms](image)

In a distributed system, time synchronization algorithms can be used to synchronize time of an each member of the system with respect to an external time reference (external time synchronization) or to synchronize time among members of the system (internal time synchronization). In case of the external time synchronization, a clock source shows *real time*, the time of international time standard, and the goal for all members of the system is to be as close to this time as possible. In case of the internal time synchronization, usually access to the *real time* is not available from within the system and the goal is to minimize time difference between any two members of the system.
Time synchronization algorithms can be implemented in hardware or in software. The hardware time synchronization can achieve very high accuracy, in the order of nanoseconds [66], but it requires a special synchronization hardware at each member of the distributed system. The software time synchronization algorithms typically use standard communication interfaces and synchronization messages to synchronize local times [67].

Time synchronization algorithms can be implemented as deterministic, probabilistic, and statistical algorithms. Deterministic algorithms assume that there is a maximum transmission delay. Using this assumption they calculate a precision that represents a maximum difference between any two clocks in the system. On the other side, probabilistic and statistical algorithms don’t make any assumption on maximum transmission delay. The precision for these algorithms is calculated using probabilistic and statistical algorithms.

3.2 Distributed Physiological Monitoring on Tier 1 of the mHealth

One or more subjects can be simultaneously monitored using multiple sensors that form Wireless Body Area Networks on Tier 1. WBAN represents a set of tiny computing elements wirelessly connected together to provide some type of health monitoring. In order to appropriately aggregate information in the WBAN, all sensors have to have precisely synchronized time. The following section will presents some exiting approaches for time synchronization in WBAN.

3.2.1 Time Synchronization in WBAN

In previous section we saw that time synchronization can be external or internal depending on a time reference. Although the external time synchronization is possible in WBAN, it is not typically used because of the limited capabilities of devices used as nodes of the WBAN. More popular is the internal time synchronization, which tends to minimize a time difference between any two members of the system. It can be realized as a master–slave or a peer-to-peer synchronization. In the master–slave configuration one node is declared as the master and all the
other nodes as slaves. The slave nodes use local time of the master as the reference time and tend to synchronize their time with the master. In the peer-to-peer configuration any node can communicate with every other node in the network. The peer-to-peer configuration offers more flexibility and eliminates the risk of the master node failure, but it is not scalable and very difficult to control [68]. There are many implementations of the time synchronization protocols; we will present a few examples in the following sections.

**Reference Broadcast Synchronization Protocol**

Example of a protocol that uses peer-to-peer configuration is Reference Broadcast Synchronization (RBS) protocol. In RBS, nodes send reference beacons to their neighbors using physical-layer broadcasts. A reference broadcast message does not contain an explicit timestamp, because receivers use only its arrival time to compare their clocks [69]. Reported accuracy of the algorithm for two nodes on Mica platform is 29.1μs [70].

RBS requires a network with a physical broadcast channel, so it cannot be used in a point-to-point communication. It also makes a lot of additional traffic with broadcast messages. The large number of messages can also lead to long convergence time.

**Timing-sync Protocol for Sensor Networks**

Timing-sync Protocol for Sensor Networks (TPSN) uses a tree structure to organize the network topology. It works in two phases: (i) in the first step the algorithm establish a hierarchical structure in the network (level discovery phase), and (ii) performs a pair-wise synchronization along the edges of this structure to establish a global timescale throughout the network (synchronization phase) [70]. This protocol gains an additional accuracy over RBS due to the multiple time-stamping of the radio messages and averaging of those time-stamps. According to authors it can achieve accuracy of 16.9μs on Mica platform for two nodes [70]. Disadvantage of the TPSN protocol is that the two-way communication prohibits the use of message broadcasting, which results in higher communication load.
Flooding Time Synchronization Protocol

Flooding Time Synchronization Protocol (FTSP) is designed for resource limited wireless platforms that require precise time synchronization. It uses low communication bandwidth and it is robust against node and link failures. Precision is achieved by utilizing MAC-layer time-stamping and comprehensive error compensation including clock skew estimation. Reported average error of the algorithm using two nodes is 1.48μs on Mica2 platform [71] and 61μs on TelosB platform [72].

3.2.2 Implementation of Distributed Wearable Monitoring on the Tier 1

Avatar system [42] represents one of the first hybrid implementations of the body sensor network. The system integrates optical, wired, and wireless communication and monitoring to achieve unobtrusive, real-time body position monitoring for real-time control of Avatars in the virtual space. A network of wearable inertial sensors (iSense) communicates through wired serial bus with the master controller that communicates wirelessly with the workstation. Since inertial sensors cannot provide absolute spatial location, optical triangulation has been used to determine absolute position of the single (master) node. Positions of inertial nodes can be reconstructed relative to the master node.

In the first implementation, a user wears three inertial iSense sensors placed on the hand, forearm, and upper arm, which are connected to master iControl sensor using wired bus, as represented in Figure 26. The iControl sensor is placed on the user’s shoulder, and equipped with the infrared LED (Light-Emitting Diode). Two Wii remotes are strategically placed in the monitored environment and used to locate IR LED on the user’s shoulder by triangulation. The iControl collects inertial data from the iSense nodes and sends it to the workstation using Bluetooth. Jovanov et al. [42] designed original system for the single user, and in case of multiple users system was not able to distinguish absolute position of their master controllers and associate their position. We extended the Avatar system to support multiple users, using the same hardware.
configuration and supporting time division multiplexing of the synchronized WBANs [73]. We implemented time synchronization and multiplexing of infrared sensors, controlled by the workstation. System organization of the proposed multi-user system Avatar+ is presented in Figure 26.

![Figure 26 System organization of the multi user system Avatar+. Time division multiplexing of synchronized controllers.](image)

The system still features two Wii remotes for absolute positioning, but activation of infrared LEDs of individual controllers is time synchronized, so only LED of one user is active at the time. Individual LEDs are symbolically presented with red and blue color in Figure 26. Symbolic representation of detected positions of individual LEDs on Wii remotes is presented as red and blue dots, accordingly.

Wii remote features infrared imaging sensors and processor that can track up to 4 moving objects. Infrared imaging sensor has resolution 128x96 pixels. A built-in image processing improves resolution by using 8x subpixel analysis to provide 1024x768 resolution for the tracked points. The sensor provides location information with 100Hz update rate, and features 33 degree horizontal and 23 degrees vertical field of view. Although Wii remote can track up to 4 points, in
case of multiple users the system cannot associate points with users. We proposed an original solution to activate LEDs only on one user at the time. Since the control program determines time slot associations with a single active user in every time slot, we can decode signals from the imaging sensors in each sampling interval and associate triangulated position to a particular user.

To allow efficient “decoupling” of tracking points from user to user we introduce idle time slot between activation of individual LEDs, as represented in Figure 26. Since the update interval of Wii remote is 10 ms, we introduce “inactive” time slot of 10 ms between individual activations of sensors.

Personal controllers are synchronized through wireless (Bluetooth) connection on iControl module that controls other inertial sensors embedded in clothes of each user. The control application running on PC was developed in C# in .NET framework. We implemented control application with multiple threads to improve real-time performance and reduce time-jitter.

We measured typical latency and jitter of time synchronization between two nodes controlled by a workstation. Our primary objective was portable system that can be easily deployed. Therefore, for our testing environment we used Toshiba laptop of 3GB RAM, Intel U4100 processor running at 1.3GHz, Windows 7 Home Premium edition, with USB CSR Bluetooth interface. A snapshot of time synchronization of controllers running at maximum switching frequency of 20Hz is presented in Figure 27. Time jitter of active periods in both controllers can be clearly seen. We evaluated the performance of the system at switching frequency of 5, 10 and 20 Hz. The results of performance analysis are presented in Table 1. All performance measurements are collected during 5 minute tests.
Figure 27 Time division multiplexing for two users at maximum frequency (50Hz); IR LED is active when the signal is low.

For all three switching frequencies nominal idle time is set to 10ms (update time of the Wii remote). Therefore, expected active time at switching frequency of 5Hz would be 200ms – 10 ms = 190 ms.

The average idle time is relative stable and independent from sampling frequency (17 – 19ms). The most critical parameter is overlap between active times of LEDs on controllers. This overlap is possible because of unpredictable delay through multiple layers of software and wireless communication. The maximum overlap between active periods caused by time jitter is 3-5 ms that is significantly smaller than the update rate of Wii remote. The performance of the system could be improved using a dedicated real-time control system with precise control of tasks. However, our main objective was optimum performance on off-the-shelf platform and standard operating system.

Table 1 Real-time synchronization performance.

<table>
<thead>
<tr>
<th>Frequency [Hz]</th>
<th>Average idle time [ms]</th>
<th>Maximum overlap [ms]</th>
<th>Expected active time [ms]</th>
<th>Measured active time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>19.2</td>
<td>4.8</td>
<td>190</td>
<td>110 – 195</td>
</tr>
<tr>
<td>10</td>
<td>18.5</td>
<td>4.4</td>
<td>90</td>
<td>45 – 95</td>
</tr>
<tr>
<td>20</td>
<td>17.4</td>
<td>3.2</td>
<td>40</td>
<td>20 - 55</td>
</tr>
</tbody>
</table>
3.3 Distributed Physiological Monitoring on Tier 2 of the mHealth

Multiple subjects can be simultaneously monitored using multiple personal devices on Tier 2. Recent proliferation and wide adoption of the smartphones make them the first choice for the implementation of the personal devices. In the case of experiments with simultaneous monitoring of multiple subjects a precise time synchronization of personal devices is necessary. The following section will focus on the time synchronization mechanisms for smartphones. Most of the mechanisms that will be described in the next section are applicable to personal computers as well.

3.3.1 Time Synchronization of Personal Devices

Modern smartphones usually offer several wireless communication interfaces (Bluetooth, WiFi, 3G/4G, GPS, etc.) that can be used for time synchronization, but applicability of specific interface and its performance will be application dependent. In order to comprehensively explore design space of possible time synchronization algorithms implementations, the algorithms will be presented in two categories according to a time reference external and internal.

External Time Synchronization

In case of the external time synchronization, all nodes (smartphones) of a distributed system synchronize their time to some external reference. For this type of synchronization we can use GPS, 3G/4G, and 802.11 b, g, n (WiFi) communication interfaces.

External Time Synchronization Using GPS

Typical time reference for external time synchronization in a distributed wireless sensor network is Global Positioning System (GPS). GPS consists of 32 satellites [74], each transmitting coordinated “GPS Time” according to its onboard atomic clock. It is typically used for determining location of a device on Earth, and its operation depends on very accurate time synchronization, where clocks of all of satellites are synchronized to within a few nanoseconds of
each other. The publicly available GPS (Civilian code) has a time accuracy specification of 340 ns, but it typically performs at 35 ns.

Modern smartphones usually have built-in controllers capable of receiving and processing signal from the GPS satellites. Most modern smartphone’s operating systems provide system support and standard application interface for accessing the GPS information. Therefore, smartphone applications can use GPS clock for precise time synchronization. On the other side, smartphones are usually running complex operating systems (e.g., iOS Android), which inherently creates unknown time jitter in message propagation through its layers. This jitter and granularity of smartphones OS scheduler reduce effective precision for time synchronization from (theoretically) microsecond range to several milliseconds (20 ms) or better [75].

Other limitations of using GPS-based time synchronization on smartphones are the same as limitations of GPS in time synchronization of WSN: (i) it is available only outdoors and (ii) it significantly reduces the battery life of the device.

External Time Synchronization Using 3G/4G and WiFi

Unlike GPS, which is sporadically used on smartphones, network communication interfaces (e.g., 3G/4G and WiFi) are typically used all the time, which makes them very attractive for time synchronization. The best known mechanism for time synchronization over network is Network Time Protocol (NTP) [67], [76]. It is a networking protocol for time synchronization that can work over packet-switched, variable-latency data networks. Although it is developed and commonly used for time synchronization of desktop computers and other network devices, NTP quickly became popular on smartphones as the most popular time synchronization mechanism.

For more than 25 years of existence, several versions of NTP are developed. The latest NTPv4 offers several enhancements over the previous versions of NTP. The most important enhancement is improved accuracy to tens of microseconds in local area networks [67]. NTPv4
can usually maintain time to within 10-20 milliseconds over the public Internet [67]. Typical external reference for time synchronization using NTP is one of the publicly available NTP servers. The *pool.ntp.org* is an example for virtual cluster of timeservers providing an easy and convenient access to NTP services [77].

Requirements for external NTP time synchronization is Internet access to NTP servers. Since smartphones are designed to work in environment with internet access, NTP is very suitable solution for time synchronization.

**Internal Time Synchronization**

Whereas the external time synchronization represents a centralized process, the internal synchronization is a cooperative activity among all the members of a system. Typically, one node (smartphone) of the distributed system will keep reference time as master controller, and all other nodes will synchronize their time with the master. The master can be predefined, or it can be elected by members of the distributed system. Both typical wireless communication interfaces of a smartphone, WiFi and Bluetooth, can be used for the internal time synchronization.

**Internal Time Synchronization Using WiFi**

In case of the internal synchronization, WiFi communication interface can be used in two configurations. In the first configuration, one smartphone can create wireless access point (or independent wireless access point can be used), while all other smartphones will be clients connected to it. This configuration is suitable for NTP time synchronization, where one smartphone can run internal NTP server, providing time reference and synchronization services to all other devices. Under ideal condition NTPv4 can achieve microsecond accuracy or better in local area networks, but in case of smartphone implementation this will be in the order of milliseconds [67].

The second configuration is a decentralized type of wireless network, also known as ad-hoc network. The network is ad hoc because it does not rely on a preexisting infrastructure, such
as wireless access points. Instead, each node participates in routing by forwarding data for other nodes, where forwarding of data dynamically adapts to the network connectivity. In addition to the classic Internet routing mechanisms, ad hoc networks can use flooding for forwarding of incoming messages.

Using Precision Time Protocol (PTP), reported accuracy of the time synchronization in ad-hoc network of smartphones is 0.1ms [78]. A typical range of WiFi on smartphones is around 30 m indoors and 90 m outdoors. This limits distance between members (smartphones) of the distributed system. Furthermore, reliable connectivity requires even shorter range between individual devices. This can be an important issue for some applications.

**Internal Time Synchronization Using Bluetooth**

Bluetooth is a wireless communication standard for data exchange over short distances in personal area networks (PANs). Since it is widely adopted and supported on most smartphones, it can be used for internal time synchronization of distributed smartphone applications. Several algorithms were used for time synchronization of smartphones over Bluetooth, such as Timing-sync Protocol for Sensor Networks (TPSN). In this implementation, the clock offset of a pair of phones is calculated based on the roundtrip time stamps of time-sync packets. The algorithm was implemented on two paired Android phones, and achieved accuracy was 15ms [79].

The main limitation of Bluetooth for time synchronization is its range. Typical range for Bluetooth in smartphones is 10 m, thus limiting distance between members of distributed system. Also, Bluetooth is mostly intended for direct communication between two devices and supports up to seven devices in the network, which is not scalable. Summary of possible time synchronization algorithms and their characteristic on smartphone is presented in Table 2.
Table 2 Summary of possible time synchronization algorithms and their characteristic on smartphone

<table>
<thead>
<tr>
<th>Type of time synchronization</th>
<th>Time Reference</th>
<th>Maximum Distance</th>
<th>Expected Accuracy</th>
<th>Easy Scalable</th>
<th>Increase in power consumption</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPS based</td>
<td>External</td>
<td>-</td>
<td>20ms</td>
<td>Yes</td>
<td>High</td>
<td>GPS receiver, outdoors</td>
</tr>
<tr>
<td>NTP using public server</td>
<td>External</td>
<td>-</td>
<td>20ms</td>
<td>Yes</td>
<td>Low</td>
<td>Internet Access</td>
</tr>
<tr>
<td>NTP using local server</td>
<td>Internal</td>
<td>90m</td>
<td>1ms</td>
<td>Yes</td>
<td>Low</td>
<td>WiFi</td>
</tr>
<tr>
<td>PTP using WiFi ad-hoc network</td>
<td>Internal</td>
<td>90m</td>
<td>0.1ms</td>
<td>No</td>
<td>Moderate</td>
<td>WiFi ad-hoc</td>
</tr>
<tr>
<td>TPSN using Bluetooth</td>
<td>Internal</td>
<td>10m</td>
<td>15ms</td>
<td>No</td>
<td>Moderate</td>
<td>Bluetooth</td>
</tr>
</tbody>
</table>

3.3.2 Implementation of Distributed Wearable Monitoring on Tier 2

We implemented simultaneous monitoring of multiple subjects using our custom Android application *imWell* and commercially available physiological monitor Zephyr BioHarness. Each subject is equipped with the physiological monitor and a personal device (Android smartphone with *imWell* application) as shown in Figure 28.

The BioHarness physiological monitors are capable of maintaining local time, which is associated with recorded signals and sent to the personal devices in each data packet. Local time of the physiological monitor can be also independently read and initialized from the personal device using specific communication protocol defined by Zephyr Technology. In order to synchronize local times of each physiological monitor, the personal monitor fetch global time using NTP synchronization protocol and then sends it to the physiological monitor at the beginning of the communication session. Optional resynchronization can be scheduled continuously after predefined interval of time until the end of communication session between personal device and the physiological monitor.
Figure 28 System organization for simultaneous monitoring of multiple subjects using personal devices

The system has been implemented and its performance validated. Validation of the proposed approach is performed using two smartphones and two Zephyr physiological monitors. Two sensors are taped together and programmed to send raw accelerometer data as illustrated in Figure 29.

Figure 29 Validation setup for time synchronization on the tier 2

In order to better simulate distributed environment in complex Internet environment, one personal device was connected to the Internet using 4G cellular network, while the other device was connected to the Internet using WiFi local area network. Furthermore, local times of the
personal devices were intentionally set to different values, to test accuracy of the external time synchronization, as it is shown in Figure 29.

Physiological monitors collect raw 3D acceleration data with sampling frequency of 50 Hz. Waveforms of sensors’ X acceleration axes for both monitors recorded and synchronized during validation are shown in Figure 30. In ideal case the waveforms should perfectly overlap, but we can see that one signal is slightly delayed (approximately 20 ms). Since both sensors record the same motion of taped sensors, this time difference represents precision of the time synchronization.

![Graph showing waveforms for X axis of acceleration for two Zephyr BioHarness physiological monitors during validation process](image)

*Figure 30 Waveforms for X axis of acceleration for two Zephyr BioHarness physiological monitors during validation process*

We repeated validation process 10 times and measured time difference between the two recorded signals. Results are shown in Figure 31. On average, time difference was 361 ms with standard deviation of 293 ms. Maximal time difference was 814 ms, while minimal was 23 ms. Please note that although both personal monitors were close to each other, the results would’ve been very similar for extremely distributed configurations, such as physiological monitoring of subjects on different continents. Our results demonstrate feasibility of global monitoring across the planet with sub second time synchronization.
The main obstacle in achieving better time synchronization using Zephyr’s physiological monitors is API of the physiological monitor. Although the time value on the physiological monitor can be set in milliseconds, its internal time can be read with time precision in seconds.

3.4 Distributed Physiological Monitoring on Tier 3 of the mHealth

The Tier 3 of the mHealth architecture is designed to aggregate physiological records from multiple subjects as well as records from multiple sessions of the same subject. In case of the simultaneous monitoring of large group of subjects, the Tier 3 can experience significant overload and become the main bottleneck of the whole system. Distributed implementation of mHealth server can be very helpful in overcoming this issue, especially if the monitored group is geographically distributed.

Precise information about the time when the record is captured is important, and all records in the physiological database should contain timestamps regardless of database implementation. In case of simultaneous monitoring of multiple subjects, precise timestamping allows time alignment of the physiological records. This time alignment can help researchers in studying influence of environmental and other factors on group of the simultaneously monitored subjects. Typical examples include synchronized monitoring of the football team during training, or study of effects of solar storms on human health.
3.4.1 Time Synchronization on the Tier 3

In order to preserve time order of the records, each record stored in mHealth Database contains precise timestamp. Field \textit{UTC\_time} in the table \textit{records}, Figure 32, contains precise timestamp of the record in the UTC format. The field is designed as \textit{TIMESTAMP} type, and supported precision in microseconds (requires MySQL 5.6.4 or later [80]).

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{records.png}
\caption{Structure of the Records table in the mHealth Database}
\end{figure}

Considering that records aggregated in mHealth Database can be recorded in different time zone, and in order to preserve information about local time of the record, mHealth Database contains additional field \textit{FS\_timestamp} that represent local time, as shown in Figure 32. The \textit{FS\_timestamp} field is designed as \textit{DATETIME} type in order to provide human readable time format.

3.4.2 Support for Distributed Implementation.

The Tier 3 of the mHealth infrastructure can also be implemented in distributed fashion. The mHealth Web API and mHealth Database can be implemented on different servers. Moreover, mHealth Database can be distributed on several servers. Several reasons can favor distributed implementation of mHealth Database:

- Load can be better distributed in case of large number of users,
- Network delay can be reduced by placing servers closer to the group of the subjects,
• Server can be placed inside of closed network.

Since the mHealth servers are designed to run as virtual machines, distributed deployment can be easily achieved. Moreover, the database design directly supports automatic data synchronization. Each table with physiological record contains filed that identify mHealth server where it was originally uploaded, field $ID_{mHealth\_Server}$ in Figure 32. Using built-in data replication in the database management system (DBMS), data can be automatically synchronized into the master database in predefined time intervals. In the master database, the $ID_{mHealth\_Server}$ filed is used in combination with original primary key of a table to uniquely identify each record.
Chapter Four

Power-efficient Wearable Monitoring

Energy efficiency has become the major design requirement for a range of computer platforms, from sensors, to smartphones, even servers in data centers. Energy efficiency in mobile devices is required because of the several key factors, including (i) limited energy capacity of batteries, (ii) cost considerations favoring less expensive packaging, and (iii) user convenience favoring lightweight designs with small form factors that operate for long periods without battery recharges. Energy efficiency in data centers is primary driven by the operating cost of used electrical energy.

System designers must understand dynamic power profile of system components to successfully improve energy efficiency and to reduce energy consumption. Power measurements can identify critical components or operations. Unfortunately, collection of power traces mostly relies on manual control and post-processing to synchronize power traces with events in profiled programs. In order to save time and effort, and enable accurate and fast dynamic profiling of running programs we developed an automated power measurement environment [81].

4.1 Environment for Automated Power Measurements

We developed an automated power measurements environment targeting mobile computing platforms typical for the Tier 2 of the mHealth infrastructure. The same environment can be used for the Tiers 1 and 3. The environment relies on minimally invasive instrumentation of a mobile platform using a shunt resistor on the power line and an inexpensive data acquisition system (DAQ) for sampling the voltage drop at the shunt resistor. The sampled voltage is directly
proportional to the current drawn by the platform, which in turn can be used to determine
dynamic power and total energy consumed. To provide automated capturing of power traces of
programs running on the mobile platform, we developed a custom program called
\textit{mPowerProfile}. This program runs on a workstation and interfaces both the mobile platform
through a serial link and the DAQ through a USB link, as it is shown in Figure 33, thus allowing
a user to configure the measurement setup and to synchronize the program execution on the
mobile platform with runtime power measurements. The current samples are logged into a file for
further processing and analysis, and the energy consumed is calculated and logged.

\textbf{Experimental Setup}

Figure 33 illustrates experimental setup for capturing power traces and measuring the
energy consumed during program execution on a mobile development platform. The mobile
platform is connected to a power supply ($V_{\text{SUPPLY}}$) via a low-resistance shunt resistor ($R = 0.1 \, \Omega$). The voltage drop over the shunt resistor is directly proportional to the current drawn by the
mobile platform ($V_{\text{SHUNT}} = R_{\text{SHUNT}} \times I$). The voltage is sampled using a data acquisition system
(DAQ) connected to a development workstation. The dynamic current, $I$, can be calculated from
the voltage samples from the shunt resistor as $I = V_{\text{SHUNT}} / R$.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig33.png}
\caption{Experimental setup for automated power measurements}
\end{figure}

The number of samples during the execution of a program is $n = T \times F_S$, where $T$ is the
execution time for the given program and $F_S$ is the sampling frequency. The total energy
consumed (ET) is calculated as a function of the measured current samples $I_j$ as follows:
\[ ET = \sum_{j=1}^{n} I_j \cdot V_{PLATFORM,j} \cdot \Delta t \]  

where, \( \Delta t = 1/FS \), and \( V_{PLATFORM,j} = V_{SUPPLY} - I_j \cdot R \). Note that the calculation can be simplified by assuming \( V_{PLATFORM} \) to be constant because the voltage drop over the shunt resistor is negligible.

In addition to \( ET \), we also calculate the energy overhead of the executing program alone, \( EO \), which excludes the energy needed to run the platform when idle. This energy overhead is calculated as:

\[ EO = ET - I_{idle} \cdot V_{PLATFORM,idle} \cdot T \]  

where \( V_{PLATFORM,idle} = V_{SUPPLY} - I_{idle} \cdot R \).

The accuracy of the energy estimation increases with increasing sampling frequency. The maximum sampling frequency supported by the DAQ in our setup is 200,000 samples per second (200 Ksps). For a processor core on a mobile platform running at 1 GHz we can sample the voltage every 5,000 CPU clock cycles. We experimented with different sampling frequencies in the range of 10 Ksps to 200 Ksps and evaluated their impact on the accuracy of energy calculations. We found that the energy calculated using 20 Ksps is within 1% of the energy calculated using 200 Ksps, so for our experiments we use a sampling frequency of 20 Ksps.

**mPowerProfile**

mPowerProfile is a software tool for automated capture of power traces and evaluation of energy-efficiency of programs running on mobile computing platforms. mPowerProfile runs on a workstation and it controls both the system under test (via a serial link terminal) and the DAQ (via a USB port).
Figure 34 shows the mPowerProfile’s GUI. It can operate in one of the two modes – *manually controlled measurements* and *automated measurements*. Regardless of the mode, the user first configures the DAQ channel parameters, including device and channel name, minimum and maximum voltages, wiring configuration (differential or single-ended), the number of channels (multiple channels can be sampled simultaneously), the sampling frequency, and a scaling parameter (all logged samples are multiplied by this parameter).

In the manual mode, the user selects the format (text or binary) and location of the output files where samples are to be recorded, and starts sampling by activating the Start button. Similarly, the capturing of samples is stopped by activating the Stop button.

In the automated mode, the user can capture power traces for a number of programs automatically. The user first connects to the platform under test via a serial link by specifying serial port, login parameters, and delay parameters (*Start*, *Stop*, and *Call* delays). The user also prepares a script in the script window by entering the shell commands for running applications which needs to be profiled for power. Each shell command is preceded by the number sign (‘#’) followed by the file name where the samples are to be stored. When the user activates the Start
Script button, \textit{mPowerProfile} takes the control and executes the commands from the script window. It starts capturing and logging samples from the DAQ immediately and waits for the Start delay to expire (e.g., 4000 ms) before issuing the first command over the serial link that will run an application of interest. \textit{mPowerProfile} continues sampling during the application execution as well as during the period of time determined by the \textit{Stop} delay after the application is completed. The collection of samples is then terminated and the log file is closed. The samples collected during the quiet period before the application is launched can be used to determine the platform’s idle current, $I_{\text{idle}}$. The total energy consumed, $E_T$, and the energy overhead, $E_O$, are calculated according to equations (4-1) and (4-2). \textit{mPowerProfile} delays the processing of the next command for the amount of time specified in the \textit{Call} delay parameter before repeating the previous steps for the next command. The script can include as many shell commands as needed. This way, a number of measurements can be easily executed with minimum effort of the user.

Energy efficiency in wearable physiological monitoring should be considered on each tier of the proposed mHealth architecture. Each case will be discussed separately since each tier has its own challenges and preferred solutions.

\textbf{4.2 Power-Efficiency on Tier 1 of the mHealth}

User factors, such as wearability and battery life can be crucial for any wearable personal technology. It is necessary to increase user’s satisfaction in order to increase acceptance of wearable health monitoring systems for everyday use. Wearability is mostly determined by the size and weight of physiological monitors, which are further determined by the size and weight of batteries selected to support certain functionality for a predefined period of time. Battery life of wearable monitors depends on their battery capacity and energy consumption. Since increase of battery capacity increases size, weight, and cost of sensors, we focus on reducing energy consumption of wearable monitors.
Workflow of physiological monitors is in general composed from (i) sensing and processing of physiological signals, and (ii) communication of relevant information. Processing of physiological signals can be significantly expedited with processor optimization, which will reduce energy consumption during signal processing. On the other side, communication using some of the widely supported technologies, such as WiFi and Bluetooth, do not provide energy efficiency necessary for the wearable health monitoring systems.

The following section will discuss improvement of energy efficiency using processor customization. As a case study, we designed an optimized processor to improve performance of R-peak detection in heart activity sensor.

4.2.1 Power Optimization of Signal Processing

Recent advances in engineering complex systems-on-the-chip [82] allow designers to rapidly explore the design space and customize embedded processor cores to meet performance requirements and design constraints. Application-specific custom design of processor can have custom instructions to accelerate the most critical processing operations and reduce average power consumption. We proposed and implemented a custom design of a processor [83] using Tensilica’s Xtensa design environment [84].

As an example application we use acceleration of R-peak detection, as the most important feature of heart monitors. This application processes an input electrocardiogram signal (ECG) to determine R-peaks and time interval between two heartbeats (RR interval) using a discrete wavelet transform. Initial algorithm was developed and tuned in Matlab, and ported to Tensilica’s Xtensa processing core. Then we performed profiling of the application to identify critical sections in the code. To accelerate critical sections and reduce the total energy consumed, we introduce several custom instructions that are specifically designed for this application and integrate them into the Xtensa instruction set. We consider several design alternatives that differ in software implementation and hardware complexity. These candidate designs are then evaluated.
in a multi-dimensional design space that encompasses processor performance, hardware complexity, code size, and power consumption.

**ECG R-peak Detection Algorithm**

Electrocardiogram (ECG) represents a transthoracic voltage generated by the contractile activity of the heart and recorded at electrodes attached to the surface of the skin. ECG is one of the most important physiological signals used in a number of diagnostic procedures. The ECG is characterized by a recurrent sequence of P wave, QRS complex, T and U waves. R-peak denotes the moment of contraction of ventricles. RR-intervals or inter-beat intervals are one of the most frequently used parameters. RR-intervals are crucial in Heart Rate Variability (HRV) analysis [41]; lower variability may indicate congestive heart failure, diabetic neuropathy, and other medical conditions. Therefore, efficient R-peak detection is essential for a number of wearable health monitoring applications.

In order to extract RR-intervals from an ECG signal we need a reliable and precise algorithm to detect R peaks. A number of approaches, varying in complexity and precision, have been developed for R-peak detection [85]. In our application we focus on an R-peak detection algorithm that uses a discrete wavelet transform [86], specifically the Daubechies D4 (Figure 35, lines 1-7). The first step is to implement a band pass filter on raw ECG signal using four recursive calls of the Daubechies D4 wavelet transform as shown in Figure 35 (lines 9-12). The absolute values of the processed signal are then filtered using a threshold filtering: all samples lower than a threshold are set to zero. The threshold is set to be 15% of the maximum value in the processed signal. The processed samples are kept in a sliding window array. This array is searched for local maximums, which represent potential R-peak signals. Once the potential R-peaks are located in the processed signal, the original ECG signal is searched to precisely locate a potential R-peak. Finally, the potential R-peak is upgraded to a true R-peak if there were no other true R-peaks in the previous 200 ms.
The original R-peak detection algorithm, which is geared toward post-processing of ECG signals, is adapted for real-time implementation. We apply the wavelet transform on input vectors with 64 samples. Four rounds of the wavelet transform give four processed samples that are used in detecting R-peaks as described above. The maximum used in the threshold filtering is determined on a 5 second training period, rather than on an entire ECG signal.

\[ \begin{array}{l}
1. \quad \text{function} \ [s \ d] = D4\_\text{Transform}(S) \\
2. \quad N = \text{length}(S); \\
3. \quad s1 = S(1:2:N-1) + \sqrt{3} \times S(2:2:N); \\
4. \quad s2 = S(2:2:N) - \sqrt{3}/4 \times s1 - (\sqrt{3}-2)/4 \times [s1(N/2) \ s1(1:N/2-1)]; \\
5. \quad s = (\sqrt{3}-1)/\sqrt{2} \times [s1 - (s2(2:N/2) \ s2(1))]; \\
6. \quad d = (\sqrt{3}+1)/\sqrt{2} \times s2; \\
7. \quad \text{end}
\end{array} \]

\[ \begin{array}{l}
8. \quad \text{function} \ D4 = \text{wavelet\_trans}(S) \\
9. \quad [S1 \ D1] = D4\_\text{Transform}(S); \\
10. \quad [S2 \ D2] = D4\_\text{Transform}(S1); \\
11. \quad [S3 \ D3] = D4\_\text{Transform}(S2); \\
12. \quad [S4 \ D4] = D4\_\text{Transform}(S3); \\
13. \quad \text{end}
\end{array} \]

Figure 35 Daubechies D4 transform used in ECG processing

The real-time R-peak detection algorithm is first verified in Matlab. As an input we used a selected subset of ECG recordings from the MIT Physionet database [87]. The selected ECG recordings cover a wide range of possible ECG signals, differing in amplitude, sampling frequency, offset, and units of measure (mV/µV). Figure 36 illustrates the R-peak detection algorithm for an ECG signal from the database. The signal is sampled with sampling frequency of 750 Hz, has an offset of -6 mV, and units of measure are in millivolts. The original ECG signal (blue) is processed by the wavelet transform to get the processed signals (red). The potential R-peaks are marked with green circles, and detected R-peaks are marked with red circles.

63
Figure 36 An example of the R-peak detection algorithm; green circles represent potential R-peaks and red circles represent detected R-peaks.

Processor Customization

Application specific integrated circuits require minimal energy, but such solutions lack flexibility offered by microprocessor-based systems. On the other side, general-purpose embedded processors may offer a good performance for a broad range of applications, but they may be suboptimal for a given application. An alternative approach is to use customizable processors that allow designers rapid and risk-free configuration and customization. We use Tensilica’s Xtensa customizable processors and demonstrate how it can be customized for the R-peak detection algorithm.

The design environment allows designers to quickly profile the application software, configure the Xtensa core, and add new instructions to optimize performance. System designers can explore multiple processor configurations and architectural enhancements by making area, speed, power and code-density design tradeoffs, based on real-time feedback from the design environment. Once the design requirements are met, Xtensa Processor Generator automatically creates a tailored application specific embedded processor, including matching tool chain.

System designers can build a processor core by selecting and configuring a broad range of features, including: (a) choosing among several basic processor cores that differ in
performance and cost; (b) configuring the number and type of functional units (e.g., integer multipliers or dividers, multiply-and-accumulate units); (c) enabling special instructions (e.g., zero-loop, conditional stores); (d) selecting and configuring size and organization of instruction and data caches; (e) configuring specialized memories and buses, (f) memory protection options, and so on.

The instruction set architecture can be extended with user-defined instructions. Tensilica Instruction Extension (TIE) language [84] can be used to specify new states, register file extensions, new instructions, coprocessor extensions, and new data transfer interfaces. For new instructions the TIE language allows designers to specify schedules – pipeline stages at which instructions use input operands and produce outputs.

To evaluate effectiveness of the proposed hardware customization for the R-peak detection application, we consider five hardware/software configurations, shown in Table 3, named BASE, FLPC, FIXP, FIPM, and DW3O. As we target a low-end embedded application, we use an ultra-low power and low-complexity Xtensa 9 processor core as the BASE configuration. It includes a fully pipelined integer 32-bit multiplier, 1 kilobyte instruction cache, and 1 kilobyte data cache. With maximum frequency of 250 MHz, the estimated power consumption is only 12mW.

Table 3 Evaluated Hardware and Software Configurations

<table>
<thead>
<tr>
<th>Name</th>
<th>Hardware Extensions</th>
<th>Software Implementation</th>
<th>Data types for ECG samples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FPU</td>
<td>Coproc.</td>
<td>HW FixedP</td>
</tr>
<tr>
<td>BASE</td>
<td>√</td>
<td>×</td>
<td>√</td>
</tr>
<tr>
<td>FLPC</td>
<td>√</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>FIXP</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>FIPM</td>
<td>×</td>
<td>√</td>
<td>×</td>
</tr>
<tr>
<td>DW3O</td>
<td>×</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>

The R-peak detection application is ported from Matlab to C with minimal changes. The input ECG signal is represented as single-precision floating-point data and all wavelet operations
and the R-peak detection are carried out on this data. The BASE processor configuration does not include a floating-point co-processor, so floating-point computations are emulated in software. Profiling the BASE configuration confirms our expectation that the wavelet computation is indeed a performance bottleneck.

The second processor configuration FLPC adds a floating-point co-processor with support for instructions that can operate on single-precision floating-point numbers. We expect this configuration to significantly improve performance, at the cost of additional complexity caused by the floating-point co-processor.

The next step in design space exploration is to re-design the R-peak detection software to read and process ECG samples represented in fixed-point data formats. We opt for a 16:16 format (both, integer and fractional portions are 16-bit long), because it provides a good balance between the range and precision, without any assumptions on the scale and units of the input ECG signal. The FIXP configuration has a hardware configuration identical to BASE, but the software implementation is re-designed to read and compute on fixed-point numbers. The critical operations are multiplication, addition, and subtraction on fixed-point numbers (see lines 3-6 in Figure 35).

Profiling of the new R-peak detection application shows that about 70 percent of the execution time is spent in fixed-point multiplications. The fixed-point addition is identical to integer addition. However, fixed-point multiplication requires one integer multiplication and one addition. A custom instruction, shown in Figure 37, is added to support this operation. This hardware/software configuration is named FIPM. We expect this configuration to yield significant performance improvements at the cost of added complexity.
Finally, the last configuration DW3O adds three more characteristic instructions to accelerate typical operations found in D4 wavelet transform (lines 3-6 in Figure 35). Rather than including separate hardware resources for each of these operations, they all share a fixed-point multiplier. We expect this configuration to further improve performance at the cost of additional hardware complexity.

Results

In our experimental evaluation we compare total energy required for each hardware/software configuration assuming a 45 nm technology process. In addition we evaluate performance, code density, and hardware complexity for all five configurations. We consider execution time spent in the D4 wavelet transform and the total time spent in the R-peak detection algorithm. We also consider the code size for the wavelet transform procedure and the R-peak detection procedure. These parameters are reported by the Xtensa Xplorer design environment.

Figure 38 shows a statistical estimation for the total energy spent in the R-peak detection (including the wavelet processing), broken into dynamic and leakage energy. For each hardware/software configuration we consider two design points with 60 MHz and 250 MHz clock frequencies. We can see that the FIPM and DW3O configurations provide 4- to 5-fold reductions in the total energy required, which further underscores the benefits of the custom instructions.

```
1. operation mul { out AR outR, in AR inpR1, in AR inpR2}{}
2. {
3.   wire do_signed = 1'b1;
4.   wire cbit = 0;
5.   wire[63:0] do_const = 32768;
6.   wire[63:0] temp = TIEmul(inpR1[31:0], inpR2[31:0], do_signed);
7.   wire[63:0] temp1 = TIEadd(temp,do_const,cbit);
8.   assign outR = temp1[47:16];
9. }

Figure 37 TIE language description of fixed-point multiplier
```
The performance results of our experimental evaluation are shown in Table 4. As expected, the BASE and FIXP configurations have the worst performance, requiring ~18.4 and ~15.7 million clock cycles for the R-peak detection. The majority of time is spent in the D4 wavelet transform, 15.9 and 13.3 million of clock cycles, respectively. However, the processor core is rather small and requires 22 kilogates. The FLPC configuration with a floating-point co-processor improves performance of the wavelet processing for more than 4.4 times relative to the BASE configuration, and the performance of the R-peak detection for slightly over 3 times. However, it requires an additional 12 kilogates, for the total hardware complexity of slightly over 34 kilogates.

Table 4 Execution time, code size, and complexity for evaluated configurations

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BASE</td>
<td>15,889,359</td>
<td>18,390,999</td>
<td>6908 1140</td>
</tr>
<tr>
<td>FLPC</td>
<td>3,574,784</td>
<td>5,969,956</td>
<td>6344 1152</td>
</tr>
<tr>
<td>FIXP</td>
<td>13,321,202</td>
<td>15,705,408</td>
<td>5352 1104</td>
</tr>
<tr>
<td>FIPM</td>
<td>1,111,040</td>
<td>3,495,148</td>
<td>4984 1104</td>
</tr>
<tr>
<td>DW3O</td>
<td>838,656</td>
<td>3,222,862</td>
<td>3628 1104</td>
</tr>
</tbody>
</table>

The processor core with a customized fixed-point multiplication instruction, FIPM, achieves performance improvement over the BASE configuration for over 14 times in the wavelet...
transform function, and 5.2 times in the R-peak detection algorithm. This result is achieved at the cost of additional ~6 kilogates relative to the base configuration, which is one half of the complexity required by the floating-point co-processor. Finally, the extensions that support characteristic operations found the wavelet transform, DW3O, achieve the best performance, improving the wavelet transform function performance for almost 19 times relative to the BASE configuration. However, the overall R-peak performance is improved only slightly relative to the FIPM. This is not surprising, if we know that the fixed-multiplier is a shared resource, and the time spent is the wavelet transform cease to be a bottleneck at this design point. The results in Table 4 for code size show that the customized instructions used in the FIPM and DW3O configurations significantly reduce the code size in the wavelet transform procedure, whereas the size in the R-peak detection function remains fairly constant.

4.2.2 Power Optimization of Data Transfer

Wireless data transfer represents an energy expensive task, and choice of the appropriate wireless technology is very important. In order to provide power efficient operation of a wearable monitor, our choice of the wireless technology need to take into consideration application requirements, such as average application data bandwidth, maximum required data bandwidth and latency, need for alerts, etc. Application requirements must be carefully matched with the implementation technology. Typical implementations include Bluetooth implementation for low-latency applications, and ZigBee, ANT+, and Bluetooth Low Energy for low power applications [88]. Overview of the typical wireless standards used in WBAN and their energy efficiency is shown in Table 5. Presented data are based on specifications for specific currently available chips produced by Texas Instruments [89].
Table 5 Total power, air date rate, and energy/bit of the typical wireless technologies

<table>
<thead>
<tr>
<th>Wireless Technology</th>
<th>Active power [mW]</th>
<th>Air Data Rate [Mb/s]</th>
<th>Power/bit [µJ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>WiFi (802.11g) TI CC3000</td>
<td>745</td>
<td>54</td>
<td>13.8</td>
</tr>
<tr>
<td>Bluetooth Classic (v 2.1) TI CC2560-PAN1325</td>
<td>99</td>
<td>4</td>
<td>24.75</td>
</tr>
<tr>
<td>ZigBee (802.15.4) TI CC2531</td>
<td>61.5</td>
<td>0.25</td>
<td>246</td>
</tr>
<tr>
<td>ANT+ TI CC2570</td>
<td>78</td>
<td>1</td>
<td>78</td>
</tr>
<tr>
<td>Bluetooth Low Energy TI CC2541</td>
<td>53.7</td>
<td>1</td>
<td>53.7</td>
</tr>
</tbody>
</table>

The most widely used commercially available WBAN technologies include Bluetooth [90] and ZigBee [91]. Bluetooth is a mature technology, already integrated in many personal devices, such as smartphones and tablets. ZigBee is a low power wireless technology with low data rate that is suitable for applications in home automation, industrial control, and personal health care. In last few years two new wireless technologies, ANT+ [92] and Bluetooth Low Energy [90], emerged. ANT+ focused on the fitness equipment, where it became one of the dominant wireless standards. Unfortunately, it’s still not widely supported in smartphones and currently only few models of smartphones support it. On the other side, Bluetooth Low Energy rapidly establishing itself as dominant way for low-power wireless transmission in smartphones.

Although traditional wireless technologies, such as WiFi [93], may offer the most efficient power transfer per bit of information, they will also have order of magnitude higher peak current consumption, and consequently require batteries capable of providing such a peak current. Illustration of the system design space of WBAN communication system is provided in Figure 39.
A number of parameters influence choice of the wireless technology for WBAN communication [94]. Overview of the critical parameters in the system design of the WBAN communication system is presented in the Table 6.

Table 6 Critical parameters in the system design of the WBAN communication system

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average communication bandwidth</td>
<td>Influences the active communication time of wireless controllers.</td>
</tr>
<tr>
<td>Maximum communication bandwidth</td>
<td>Critical for bursts of urgent messages, and affects the maximum latency for data transmissions</td>
</tr>
<tr>
<td>Maximum power supply current</td>
<td>Determines the type, size, and weight of the battery.</td>
</tr>
<tr>
<td>Active power</td>
<td>Determines the type, size and weight of the battery, as well as the battery life.</td>
</tr>
<tr>
<td>Standby power</td>
<td>Determines the maximum battery life, as a function of the system duty cycle</td>
</tr>
<tr>
<td>Communication start up time</td>
<td>Represents the overhead and determines the efficiency of individual transmissions</td>
</tr>
<tr>
<td>Communication setup time</td>
<td>Represents time necessary to (re)establish a connection.</td>
</tr>
<tr>
<td>Standards based communication technology</td>
<td>Influences the system interoperability.</td>
</tr>
<tr>
<td>Protocol stack size and processing requirements</td>
<td>Determine memory and processor power requirements of the wireless sensor platform.</td>
</tr>
</tbody>
</table>

4.3 Power-Efficiency on Tier 2 of the mHealth

Smartphones became dominant personal devices used in everyday life for various purposes. Their availability, high processing power, and affordability makes them very suitable
for the wearable physiological monitoring. Unfortunately, smartphone applications for the wearable monitoring reduce its expected battery life. Since the smartphones are general purpose commercially available devices, hardware customization toward energy efficient wearable monitoring is not realistic. On the other side, energy efficiency can be improved on the personal application level, thus energy efficiency should be considered during the application design and development. Understanding how energy is consumed is the first step in improving energy efficiency. In other words, we need to know how much energy is consumed during different phases of the application execution.

A number of recent research studies have focused on power profiling and power estimation of mobile computing platforms. Carroll and Heiser quantified energy consumption of each component in a mobile device by performing rigorous tests and then simulating a number of usage scenarios on mobile devices [95]. Bircher and John used processor performance counters and system-specific models to estimate consumption of CPU, memory, disk, and I/O [96]. Pathak et al. used system call tracing and known observations of the system to generate models that can perform run-time power estimation with fine grained measurements [97], [98], [99].

Several prior studies focused on capturing power traces on smartphones [95]. They relied on manual control and post-processing to synchronize power traces with events in profiled programs. Our environment for automated power measurements, presented in Section 4.1, allows accurate and fast profiling of running programs on mobile platforms. The environment is particularly useful in energy optimization of mobile devices during communication with remote server using compressions. Selection of the best compression utility and its compression level will depend on several factors, and in order to determine the best configuration we need to comprehensively evaluate design space.
4.3.1 Power Optimization of Data Transfer with and without Compressions

Communication of mobile devices with a remote server in physiological monitoring is typically energy expensive. It can be implemented using several different approaches, each focused on optimizing a particular set of requirements. One approach is to upload each record or collection of records as soon as possible. This will result in minimal delay, but will also use more energy. An alternative is to buffer messages and collected data and delay uploading until a certain block of data is collected, and then upload the data set. In addition, data block can be compressed to reduce transmission time. On the other side, the compression itself will require additional energy. Therefore, a careful tradeoff between energy savings and energy investment must be made. Different compressions will result in different energy efficiency for particular conditions, and selection of the best compression utility and its compression level will depend on several factors, as described in the next section.

Experimental Setup

We evaluated influence of frequency scaling, type of compression, and compression level on energy efficiency of data transfers using our mPowerProfile environment, described in Section 4.1. For our assessment we used development board Pandaboard, shown in Figure 40, designed by Texas Instruments to support software development for smartphones and other mobile devices [100].

Pandaboard is based on a Texas Instruments system-on-a-chip (SoC) OMAP4430 [101] with 1 GB of low-power DDR2 SDRAM. The OMAP4430 SoC includes a dual-core ARM Cortex-A9 MPCore processor, a 3D graphics accelerator, and an image signal processor. We selected this development board for our assessment because a number of commercial mobile devices, such as Amazon Kindle Fire, BlackBerry Playbook, Motorola Droid RAZR, Samsung Galaxy Tab and Galaxy S II are based on this chipset.
The platform can run various mobile open-source operating systems based on the Linux kernel, including Ubuntu, Android, and MeeGo/Tizen. We used Ubuntu distribution provided by Linaro [102] because it was the most stable, and because the measurements were similar to one made on Android.

**Methods**

We evaluated energy efficiency of lossless compression utilities for wireless transfer over WiFi network. We assessed six lossless compression utilities, presented in Table 7, for supported ranges of compression levels. Our selection of the relatively fast *gzip* utility and the slower, but better compressing utility *bzip2* was based on their widespread use. The *lzop* utility is included because of its exceptionally high speed. The *xz* utility is known for its high compression ratio, slow compression, and fast decompression. Since most of today’s processors in the smartphones, including OMAP platform in our Pandaboard, are already equipped with multiple cores, we also included *pigz* and *pbzip2*, which are parallel versions of *gzip* and *bzip2*. All of these utilities operate with byte-level granularity and support a number of compression levels that allow the user to trade off speed for compression ratio. Lower levels favor speed, whereas higher levels result in better compression.
Table 7 Lossless Compression Utilities

<table>
<thead>
<tr>
<th>Utility</th>
<th>Compression levels (default)</th>
<th>Version</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>gzip</td>
<td>1-9 (6)</td>
<td>1.4</td>
<td>DEFLATE (Ziv-Lempel, Huffman)</td>
</tr>
<tr>
<td>lzop</td>
<td>1-9 (3) (2-6 equivalent)</td>
<td>1.0.3</td>
<td>LZO (Lempel-Ziv-Oberhumer)</td>
</tr>
<tr>
<td>bzip2</td>
<td>1-9 (9)</td>
<td>1.0.6</td>
<td>RLE+BWT+MTF+RLE+Huffman (100KB-900KB)</td>
</tr>
<tr>
<td>xz</td>
<td>0-9 (6)</td>
<td>5.1.0alpha</td>
<td>LZMA2</td>
</tr>
<tr>
<td>pigz</td>
<td>1-9 (6)</td>
<td>1.1.5</td>
<td>parallel implementation of gzip</td>
</tr>
<tr>
<td>pbzip2</td>
<td>1-9 (9)</td>
<td>2.1.6</td>
<td>parallel implementation of bzip2</td>
</tr>
</tbody>
</table>

For each compression type and compression level we measured energy efficiency of the Pandaboard running on four different frequencies: 300MHz, 600MHz, 800MHz, and 1.01GHz. The isolated tests are ensured by disabling any unnecessary tasks or processes prior to conducting any experimental runs on the mobile platform. This includes disconnecting unused hardware (leaving only serial and power cable connected) and turning off GUI interface (leaving only terminal interface).

Results

Although we measured total energy and energy overhead, as it is described in Section 4.1, we will present here only results for energy overhead, assuming that I_idle = 0. The reason for such decision is our preliminary findings based on additional measurement conducted on Nexus 4 [44], which suggests that smartphones, such as Nexus 4, have much lower idle current, while power profile of program execution closely resemble one recorded on the Pandaboard. Figure 41 shows energy efficiency for data compression and upload using compression utilities (EE.C) depending on compression level and operating frequency of the CPU.
Our results show that the best energy efficiency is achieved using **pigz** compression utility for level 1 compression level, running on 600MHz. In this configuration, energy efficiency is 2.5 MB/J, which results in 6.4 times improvement compared to energy efficiency of 0.39 MB/J for uncompressed data upload.

Figure 42 shows energy efficiency for data download and decompression using compression utilities (EE.D) depending on compression level and operating frequency of the CPU.
The results are indicating that the most energy efficient approach is **pigz** compression utility with level 9 compression level, running on 300MHz. In this configuration, energy efficiency is 7.35 MB/J, which results in 6.125 times improvement compared to energy efficiency of 1.2 MB/J for uncompressed data download.

### 4.4 Power-Efficiency on Tier 3 of the mHealth

Energy efficiency of mobile (portable) devices is important because of their limited amount of available energy; however, it is also important to improve energy efficiency of servers. Electricity used in global data centers in 2010 was between 1.1 and 1.5% of total electricity used. In the US, it was between 1.7 and 2.2% of total electricity used [103]. Moreover, cost of electrical energy in global data centers represents significant part of their operating costs, and according to some estimates better algorithms and smarter data processing can reduce up to 35% in total electricity used, which can result in savings for data center of $2.8 million per year [104].

In order to help improve energy efficiency in the Tier 3 of the mHealth infrastructure, we evaluated influence of frequency scaling and number of threads on energy efficiency.

**Experimental setup**

We used Dell PowerEdge T110 II platform featuring a quad-core Intel Xeon E3-1240v2 processor based on Ivy Bridge architecture. Each processor core supports 2-way multithreading; as a result, the total number of logic processor cores is eight. The CPU supports a number of frequency steps ranging from 1.60 GHz to 3.40 GHz, and features a three-level cache system, with 256 KB L1 data cache, 1 MB L2 cache, and 8 MB L3 cache. The system memory is 16 GB DIMM DDR3 synchronous at 1600 MHz. The secondary storage includes a serial ATA hard disk with capacity of 1 TB. The platform includes a gigabit network interface, a USB controller, and audio and video interfaces.

We selected this platform since it allows the use of **likwid** lightweight performance tools to perform power measurement, specifically **likwid-powermeter** [105]. The Intel Xeon E3-1240v2
processor includes -- an on-chip resource for estimating energy and power of running tasks using events recorded in performance monitoring registers and their proprietary model that captures physical characteristics of the processor. The likwid tool interfaces the power meter and outputs power measurements in joules and watts. Intel researchers demonstrated that this on-chip resource gives estimates for the energy and power that are within several percents of those acquired by the actual power measurements [106]. The platform was running a Linux operating CentOS 6.3 [107].

**Methods**

Since the processing on the server can include different types of processing, we used PARSEC benchmark suite [108] to evaluate influence of frequency scaling and number of threads on energy efficiency and execution time. PARSEC benchmark suite provides 12 different benchmarks, but we selected 6 benchmarks with functionality similar to mHealth server (data storage, search, and data mining). The list of the selected benchmarks and their application domains, and descriptions is presented in Table 8.

*Table 8 Benchmarks and their application domains used for energy efficiency estimation*

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Application Domain</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>blackscholes</td>
<td>Financial Analysis</td>
<td>Option pricing with Black-Scholes</td>
</tr>
<tr>
<td>dedup</td>
<td>Enterprise Storage</td>
<td>Next-generation compression with data deduplication</td>
</tr>
<tr>
<td>ferret</td>
<td>Similarity Search</td>
<td>Content similarity search server</td>
</tr>
<tr>
<td>freqmine</td>
<td>Data Mining</td>
<td>Frequent itemset mining</td>
</tr>
<tr>
<td>streamcluster</td>
<td>Data Mining</td>
<td>Online clustering of an input stream</td>
</tr>
<tr>
<td>swaptions</td>
<td>Financial Analysis</td>
<td>Pricing of a portfolio of swaptions</td>
</tr>
</tbody>
</table>

Each benchmark was run at 14 different CPU frequencies (3.4, 3.3, 3.1, 3, 2.8, 2.6, 2.5, 2.4, 2.2, 2.1, 2, 1.9, 1.7, and 1.6 GHz) in combination with 1 to 16 threads, which resulted in 224 runs for each benchmark.
Results

Measured values for energy consumption are converted to speedup by comparison with single thread execution at 3.4 GHz. Speedups of all aforementioned benchmarks for each frequency and thread are then combined using geometric mean. Overall improvement of energy efficiency is shown on Figure 43.

Figure 43 Energy efficiency improvement comparing to single threaded execution on 3.4GHz

Our results indicate that the most efficient processing is 8-thread execution at 2.1GHz. This configuration improves energy efficiency 2.3 times compared with single-thread execution at 3.4GHz.
Chapter Five
Implementations and Original mHealth Applications

Described mHealth architecture enables energy efficient distributed data collection, recording, and storing of physiological signals into mHealth Database. Although recording and archiving raw signals can be useful, often processing of the signals is necessary to extract relevant information from the signals. The processing of the signals can be performed after the signals are recorded, or it can be performed at the time the signals are recorded (real-time processing). While the processing of already recorded signals is typically performed on the Tier 3 of the mHealth architecture, real-time physiological monitoring is mostly performed on the Tier 2.

In case of the real-time processing, relevant information is extracted in real-time, so storing of the raw signals may not be required. This scenario leads to reduction of amount of data that has to be uploaded to Tier 3, which results in reducing required data bandwidth and energy consumption for data transmission. The real-time processing also distributes processing load across the personal devices, thus increasing scalability of the whole system. Furthermore, it enables instantaneous feedback to the user.

We developed several original applications that process signals and extract relevant information in real-time. In the following sections each of this application will be presented separately.
5.1 Real-time Assessment of Physiological Response to Posture Transitions

Cardiovascular diseases represent the leading cause of death in the United States. The current practices in monitoring health status are expensive, often insufficient, and limited to monitoring in healthcare facilities. Patients usually consult cardiologists only after experiencing first symptoms of deteriorating cardiac status. Cardiologists usually administer exercise stress test to assess cardiac status. Standard stress tests are bike or treadmill exercise supervised by a doctor or a trained technician to determine the level of exercise a patient can tolerate. These tests are usually followed by prescribed therapeutic lifestyle changes including modification of diet and development of an exercise program with a follow-up after several months. However, the clinicians do not have tools to assess the patients’ progress and their compliance to the prescribed therapies. Patients and healthy users lack tools to monitor their own cardiac status and to manage healthy lifestyle.

Posture changes create physical stress that challenges physiological homeostasis. It was hypothesized that cardiovascular response to posture changes can be used to assess the state of the cardiovascular system and overall wellness status during activities of daily living [49]. Figure 44(b) shows heart rate recorded during a posture transition from sitting to standing for a healthy subject (blue line) and a subject with cardiac condition (red line). The heart rate in the healthy subject quickly increases as a response to the physical activity and then returns to a level that corresponds to the new homeostasis.

We hypothesize that characterizing this dynamic response, including a change in the heart rate and time to reach the maximum, can be used to indicate the subject’s cardiac status and overall wellness. Long term monitoring of cardiac health and physical activity during activities of daily living can provide indication of short term and long term changes of cardiac status and fitness. We developed an inconspicuous wearable cardiac and wellness assistant called imWell (inconspicuous mobile Wellness) for efficient unobtrusive monitoring of physical activity, cardiac health, and overall physical wellness.
Figure 44 Modeling change of cardiovascular response induced by transition from sitting to standing; (a) illustration of user's body position during the transition; (b) heart rate before, during, and after the transition.

The imWell system continually records interbeat intervals and processes accelerometer data to recognize posture transitions. For detected sit-to-stand posture transitions, the system characterizes dynamic heart response by extracting transition timestamp, duration, heart rate before and after transition, and the time needed to reach the maximum heart rate from the beginning of the transition. This way we treat each sit-to-stand posture change, which happens dozens of times per day, as a simplified “stress tests” of our cardiovascular system. Analyzing the results of these tests over a long period of time could be used to assess cardiac status and fitness level of the user.
5.1.1 Related Work

Recognition and quantification of human activities using small wearable sensors during activities of daily living has been increasingly used in many applications. Automatic activity recognition and quantification systems that utilize inertial sensors are proposed for long-term health and fitness monitoring [109], [110], assessing of mobility in elderly and people with Parkinson’s disease [111], [112], [113], automatic fall detection [114], [115], and rehabilitation [116]. Approaches for automatic activity recognition used by researchers vary in number, type, and placement of utilized sensors, as well as in processing of recorded signals. While some researchers used multiple sensors for automatic activity recognition [117], [118], [119] increasing number of projects use a single inertial sensor [120], [121], [122], [123] usually placed on the chest.

One of the most challenging tasks in automatic activity recognition is detection and differentiation of posture transitions sit-to-stand and stand-to-sit. A number of researchers monitored angle of trunk tilt to detect possible posture changes by detecting situations when the angle exceeds certain threshold [120], [122], [123], [124]. While similar approaches for detection of possible posture transition are used, researchers used significantly different approaches in order to differentiate aforementioned transitions. Godfrey et al. [123] used vertical velocity, Fleury et al. [124] used wavelet-based pattern recognition on accelerometer data and tilt angle, while Fuentes at al. [122] used angle and forward and vertical acceleration.

5.1.2 Methods

Our system relies on physiological monitor Zephyr BioHarness 3 to capture both inertial and physiological data (acceleration and RR-intervals). The monitor is placed in a chest belt worn by the user and communicates the data via Bluetooth to a personal device. The personal device (a) processes the signals to detect and timestamp transitions using data from inertial sensors, (b) extracts, characterizes, and records dynamic heart response to sit-to-stand transitions, and (c) uploads the records to a mHealth server [125].
Posture Transition Detection

Figure 44 (a) shows a user during a transition from seating to standing and orientation of the Zephyr’s inertial sensor, with vertical axis (x), lateral (y), and sagittal (z). The Zephyr monitor is mounted in side strap located under subject’s left arm. The Zephyr’s inertial sensor samples and reports acceleration components $A_x$, $A_y$, and $A_z$ with the sampling frequency of 50 Hz. When the user is standing still or seating in the upright position we should observe $A_x = -1g$, $A_y = A_z = 0$, where the $g$ is the Earth gravitation (9.81 m/s²).

The first step in identifying possible posture transitions is to detect a change in the upper body angle relative to the upright position, $\theta$, and a change in the acceleration vector magnitude, AccMag. The upper body angle is calculated as follows:

$$\theta = \arctan(A_z/A_x) \tag{5-1}$$

where $\theta = 0$ degrees in the upright position as shown in Figure 44(a). It has a characteristic signature during posture transitions sit-to-stand and stand to sit. Figure 45(b) shows the angle during a sit to stand transition. We can locate 3 characteristic points in the angle signature – two purple circles mark the beginning ($\text{BeginTransition}$) and the end of the transition ($\text{EndTransition}$), and purple diamond marks the angle peak ($\text{AnglePeak}$).
Figure 45 An example of physiological response during transition from sitting to standing: a) real-time heart rate calculated from RR intervals and hip position during transition; b) angle and magnitude calculated from accelerometer signals and hip position during transition.

To identify a possible posture transition we search for a segment where the angle rises above a certain threshold. However, the angle \( \theta \) may include offsets caused by imperfect monitor placement and monitor’s movements relative to the body. To eliminate the impact of the offsets we calculate the standard deviation of the angle \( \theta \) on a sliding window of 1 second, \( stdev(\theta) \), and...
use it to locate possible posture transitions. This signal is searched for maximums in segments that are above a certain threshold. Once we locate candidate segments we search for three characteristic points in the original angle \( (\text{BeginTransition}, \text{AnglePeak}, \text{EndTransition}) \).

To estimate the level of physical activity we calculate the standard deviation of the acceleration vector magnitude on a sliding window of 1 second, \( stdev(\text{AccMag}) \). The \( stdev(\text{AccMag}) \) is used to distinguish between posture transitions and motion artifacts such as leaning forward/backward/sideways in a chair. If the maximum \( stdev(\text{AccMag}) \) in a candidate segment is higher than a certain threshold, we mark this segment as a true transition. In true transition segments, we search the acceleration vector magnitude for two characteristic points, marked by blue circles in Figure 45 (b), representing the maximum and the minimum \( (\text{AccMagMax}, \text{AccMagMin}) \).

sit2stand = 0
if \((\text{Time(AccMagMax)} < \text{Time(AccMagMin)})\)
    sit2stand++
else
    sit2stand--
    if \(((\text{Time(AccMagMax)} - \text{Time(BeginTransition)}) < (\text{Time(EndTransition)} - \text{Time(AccMagMax)}))\)
        sit2stand++
    else
        sit2stand--
        if \((\text{Time(AccMagMax)} < \text{Time(AnglePeak)})\)
            sit2stand++
        else
            sit2stand--
        if \((\text{sit2stand} > 0)\)
            Transition=Sit-To-Stand
        else
            Transition=Stand-To-Sit

\[\text{Figure 46 Algorithm for posture transition recognition}\]

The next step is to determine the type of the posture transition. In order to distinguish between the sit to stand and the stand to sit transitions, we employ multiple criteria that consider ordering of the characteristic points in time. The type of the transition is determined by a voting system described in Figure 46.
Dynamic Heart Response

Heart activity is continually recorded in our application as a stream of RR intervals. In addition, dynamic heart response to a sit-to-stand posture transition is characterized in near real-time. The characterization is performed once the transition is detected and involves determining characteristic points as shown in Figure 44 (b): (i) heart rate at the beginning of posture transition ($HR_b$), (ii) the maximum heart rate after the transition is performed ($HR_a$), and (iii) timestamp of the maximum heart rate. From these parameters we calculate the heart rate difference ($HR_{diff}$) and the time needed to reach the maximum heart rate from the beginning of the transition ($T_{HRMAX}$). To find $HR_a$, we search for the maximum heart rate in a time window of 13 seconds after EndTransition. If it is detected at the end of this window, we continue the search for the maximum beyond this point in time, until the heart rate starts decreasing.

For each sit-to-stand transition we record the following parameters:
(a) $Time(BeginTransition)$, (b) $T_{Transition} = Time(EndTransition) - Time(BeginTransition)$, (c) $HR_a$, $HR_b$, $HR_{diff}$, (d) $T_{HRMAX}$, (e) $T_{SIT}$, and (f) $T_{STAND}$.

$T_{SIT}$ is the duration of seating before the transition, and $T_{STAND}$ is duration of standing still after the transition. $T_{SIT}$ and $T_{STAND}$ can help further characterize posture transitions: e.g., in analysis we would consider only transitions for which $T_{SIT}$ is longer than a certain time period (e.g., 60 seconds) and $T_{STAND}$ is longer than $T_{HRMAX}$.

5.1.3 Smartphone Application

imWell Heart Response smartphone application implements the algorithms described in the previous sections to support monitoring and logging of dynamic heart response to posture transitions. The recorded logs of registered users can optionally be uploaded to mHealth Databases.
Figure 47 shows screenshots of the application. The user starts monitoring dynamic heart response to posture transitions by pressing the Start/Stop Recording button. During the session imWell Heart Response displays current heart rate, respiration rate, current posture, and duration of the session. It also displays the list of made sit-to-stand transitions and subset of their parameters.

5.1.4 Experimental Setup

We record signals during a series of cued posture transitions driven by our custom program [126]. The protocol includes: (1) Quiet sitting for 5 min, (2) Standing still for 30 sec, (3) Quiet sitting for 2 min, (4) Standing still for 30 sec, (4) Sitting & working on computer for 1 min, (5) Walking for 30 sec, (6) Sitting and working on computer for 1 min, (7) Walking for 30 sec, (8) Sitting and working on computer for 1 min, (9) Walking for 30 sec, and (10) Sitting and working on computer for 1 min.

In addition, we record videos of experiments using the custom program that captures video and synchronizes with other measurements and the cues, described in Section 2.4.2. The experiments with multiple participants were conducted in the Real-time Physiological Monitoring...
Lab [126] and Autonomous Tracking Optical Measurement (ATOM) Lab. The ATOM lab allows accurate tracking of reflective markers using 33 Vicon T40 series IR cameras [127]. The Vicon system records the absolute position of each marker with millimeter precision every 20 ms.

Our algorithms for detection and characterization of posture transitions rely exclusively on the data from the Zephyr monitor, and we use the Vicon system only for verification of algorithms. The experiments in the ATOM lab allow us to accurately capture body movements during posture transitions and thus precisely determine the beginning and the end of each posture transition. The users wear the Zephyr BioHarness monitor and strategically placed reflective markers.

5.1.5 Results, Discussion, and Conclusion

The inertial and heart activity data are collected on 7 healthy participants who performed the protocol described in the previous section. To verify accuracy of posture detection algorithm we compare the transitions reported by the program with actual transitions from the protocol and absolute positions and timing recorded using the Vicon system. We found that the proposed algorithm performs flawlessly recognizing correctly all sit-to-stand and stand to sit transitions correctly.

Table 9 shows the average and the standard deviation of a subset of parameters characterizing dynamic heart response to sit-to-stand transitions in the protocol for all subjects. The total averages and standards deviation for all subjects are as follow: $T_{\text{Transition}} = 2.7 \pm 0.69$ s, $HR_{\text{diff}} = 27.36 \pm 9.30$ bpm, $HR_a = 63.30 \pm 9.02$ bpm, $HR_s = 90.66 \pm 10.09$ bpm. The results show a significant change of heart rate ($HR_{\text{diff}}$) for each subject indicating potential of the proposed parameters in characterization sit-to-stand transition.
Table 9 The average and the standard deviation of a subset of parameters characterizing dynamic heart response to sit-to-stand transitions for 7 subjects (S1-S7).

<table>
<thead>
<tr>
<th>Subject</th>
<th>T_{Transition} [s]</th>
<th>HR_{b} [bpm]</th>
<th>HR_{diff} [bpm]</th>
<th>T_{HRMAX} [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>2.97±0.46</td>
<td>66.25±5.36</td>
<td>36.75±5.01</td>
<td>11.90±2.39</td>
</tr>
<tr>
<td>S2</td>
<td>2.73±0.63</td>
<td>81.2±4.82</td>
<td>18.79±4.41</td>
<td>8.99±2.08</td>
</tr>
<tr>
<td>S3</td>
<td>2.32±0.48</td>
<td>59.18±5.55</td>
<td>37.23±5.44</td>
<td>13.64±2.46</td>
</tr>
<tr>
<td>S4</td>
<td>1.99±0.74</td>
<td>53.66±4.23</td>
<td>32.73±6.33</td>
<td>12.16±3.62</td>
</tr>
<tr>
<td>S5</td>
<td>2.57±0.43</td>
<td>58.78±0.69</td>
<td>25.00±4.48</td>
<td>13.05±2.49</td>
</tr>
<tr>
<td>S6</td>
<td>2.74±0.47</td>
<td>61.12±2.53</td>
<td>26.51±3.53</td>
<td>9.22±4.58</td>
</tr>
<tr>
<td>S7</td>
<td>3.61±0.50</td>
<td>62.90±1.73</td>
<td>14.48±3.66</td>
<td>14.01±2.58</td>
</tr>
</tbody>
</table>

Whereas the goal of this processing was to develop the enabling technology for unobtrusive capturing of heart dynamic response to posture transitions, further research is needed to develop methods to assess cardiac status and fitness using the proposed parameter and to provide feedback and guidance to the users.

5.2 Real-time Quantification of Timed-Up-and-Go Test

The Timed-Up-and-Go (TUG) is a frequently used clinical test for assessing balance, mobility, and fall risk in elderly population and people with Parkinson's disease [51]. It is simple and easy to administer in an office, and thus can be used in screening protocols. The test measures the time a person takes to perform the following tasks: rise from a chair, walk three meters, turn around, walk back to the chair, and sit down. Longer TUG times have been associated with mobility impairments and increased fall risks [51], [52], [53]. Adults without balance problems can perform this test in less than 10 seconds. Alternatively, adults with mobility difficulty may require more than 30 seconds. TUG duration is also sensitive to therapeutic interventions, e.g., in Parkinson’s patients [128], [129]. Whereas the test has been proven valuable in early assessment of balance and mobility, it is limited as its only outcome is the time to complete the test.

An instrumented Timed-Up-and-Go (iTUG) test has been recently introduced [130], [131]. In this test, the subject is instrumented by a dedicated device specially designed for gait...
and movement analysis. A number of additional parameters can be derived that can better indicate gait and balance impairments, including Sit-to-Stand duration, Stand-to-Sit duration, the amplitude range of anterior-posterior acceleration, etc. iTUG has proven to be sensitive to pathologies [130], [131] and useful in fall risk prediction [132].

We developed a smartphone application called sTUG that completely automates the iTUG test so it can be performed at home. sTUG captures the subject’s movements utilizing smartphone’s built-in accelerometer and gyroscope sensors, determines the beginning and the end of the test and quantifies its individual phases, and optionally uploads test descriptors into a medical database.

We believe that this application could be of great interest for older individuals and Parkinson’s disease patients as well as for healthcare professionals. The procedure requires minimum setup (chair and marked distance of 3 meters) and inexpensive instrumentation (smartphone placed on the chest or belt running the sTUG application). The feedback is provided instantaneously to the user in a form of a report with the values of all significant parameters that characterize the TUG test. It is easy to use and users can take multiple tests in a single day at home (e.g., to assess the effects of drugs). With automatic updates to the mHealth server, caregivers and healthcare professionals can gain insights into overall wellness of the subjects. For example, they can assess the impact of therapeutic interventions (e.g., impact of drugs) by analyzing the parameters from multiple tests performed in a single day. Next, healthcare professional and researchers can monitor and evaluate evolution of disease by analyzing the trends in the parameters collected over longer periods of time.

5.2.1 Related Work

The prior TUG studies utilized either specialized devices for movement analysis (e.g., McRoberts Dynaport Hybrid) or custom inertial sensors (e.g., accelerometers) that were mounted on the subject’s lower back. Such devices typically include a 3-dimensional accelerometer and
can record x, y, and z acceleration components during the TUG test. The data are later analyzed off-line to parameterize the TUG test. Mellone et al evaluated suitability of a smartphone’s built-in accelerometer for the iTUG [115]. They compared the subject’s anteroposterior acceleration measured concurrently on a smartphone and a state-of-the-art device for movement analysis, and found the statistical agreement between the two. However, this study did not pursue a more ambitious goal of using smartphone application for quantifying the TUG test.

Recognition and quantification of human activities using small wearable sensors during activities of daily living has been increasingly used in many applications. Automatic activity recognition and quantification systems that utilize inertial sensors are proposed for long-term health and fitness monitoring [109], [110], assessing of mobility in elderly and people with Parkinson’s disease [111], [112], [113], automatic fall detection [114], [115], and rehabilitation [116]. Approaches for automatic activity recognition used by researchers vary in number, type, and placement of utilized sensors, as well as in processing of recorded signals. While some researchers used multiple sensors for automatic activity recognition [117], [118], [119] increasing number of projects use a single inertial sensor [120], [121], [122], [123] usually placed on the chest.

5.2.2 Methods

Great advantage of the sTUG application is that utilize only smartphone’s built-in sensors. A subject mounts the smartphone on his/her chest or belt and starts the application. The application records and processes the signals from the smartphone’s gyroscope and accelerometer sensors to extract the following parameters that quantify individual phases of the iTUG: (a) the total duration of the TUG, (b) the total duration of the sit-to-stand transition, and (c) the total duration of the stand-to-sit transition. In addition, we extract parameters that further quantify body movements during sit-to-stand and stand-to-sit transitions, including the duration of sub
phases, maximum angular velocities and upper trunk angles. These parameters are recorded on
the smartphone and optionally uploaded to a mHealth server.

**Body Instrumentation**

Figure 48 illustrates typical phases of the iTUG test that involves a sit-to-stand transition,
walking forward for 3 meters, turning around a cone, walking back to the chair, and a stand-to-sit
transition. We instrument a subject by placing a smartphone on his or her chest. The smartphone
is placed in a holder that is mounted on two elastic textile straps, so it remains fixed relative to the
trunk position during a test. Whereas this setup may pose challenges to some of the subjects who
are unable to tie a strap over their chest without assistance, alternative attachments are possible.
For example, the smartphone may be placed in a textile holder that is worn around the neck and
attached to the clothes using a Velcro strap. Alternative setups are possible with a smartphone
placed in a belt or in a holder on the hip. However, we assume that the smartphone is placed close
to sternum in an upright position or clipped on the front side of a waist belt.

![Figure 48 iTUG test phases and smartphone instrumentation of the subject](image)

Figure 49 shows the smartphone orientation: the smartphone’s z axis corresponds to the
sagittal axis, the y corresponds to the vertical axis, and the x corresponds to the frontal axis of the
human body.
Modern smartphones typically integrate a number of inertial and navigation sensors including an accelerometer, a gyroscope, and a magnetic sensor. The accelerometer measures proper acceleration and is typically used to keep the screen upright regardless of the smartphone orientation. With the proposed mounting of the smartphone, we record the $x$, $y$, and $z$ components of the acceleration of the subject’s upper trunk and use these signals to determine upper body position. The gyroscope measures angular movements, specifically the rotation around the $x$ axis (a.k.a. roll), $y$ axis (yaw), and $z$ axis (pitch). In our setup, we record the angular speed from the gyroscope and use the signals to detect and locate important transitions during the iTUG test. The magnetic sensor is sensing the Earth magnetic field and is normally used to aid navigation by determining the Earth’s magnetic poles and the smartphone’s orientation. In our setup we can use the signal to detect changes in direction during walking.

**iTUG Characterization**

To characterize the TUG test we start from a list of parameters suggested by Weiss et. al [130]. We expand this list to include new parameters and to refine the definitions of the original parameters that were lacking due to limitations by the measuring setup used in [130]. Table 10 lists the proposed parameters and describe their meaning and metric units. In addition to the total duration of the test ($d.TUG$), we determine the total duration of the sit-to-stand transition...
To better characterize the sit-to-stand transition we determine its two phases – a leaning forward phase \((d.LF)\) and lifting up phase \((d.LT)\), the maximum trunk angle \((a.S2ST)\) during the lean forward phase, the maximum angular velocities in the lean forward \((v.LF)\) and the lift up phase \((v.LT)\). Similarly, we determine duration of separate phases of the stand-to-sit transition, a preparing to sit phase \((d.PS)\) and a sitting down phase \((d.SD)\).

Figure 50 shows raw acceleration and gyroscope data recorded on a smartphone mounted on a chest of an individual with diagnosed Parkinson’s disease during a TUG test. The top graph shows the \(x\), \(y\), and \(z\) components of the acceleration measured in \(m/s^2\), and the bottom one shows the angular velocity measured in \(\text{radian/s}\). The data are recorded using our custom Android application running on a Nexus 4 smartphone. The sampling frequency is set to 100 Hz. The following subsection describes algorithms that process the raw sensor data to extract the described parameters as shown in Figure 50.

**Table 10 Parameters for iTUG Characterization**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>d.TUG</td>
<td>Total duration of the TUG test</td>
<td>seconds</td>
</tr>
<tr>
<td>d.S2ST</td>
<td>Total duration of the sit-to-stand transition</td>
<td>seconds</td>
</tr>
<tr>
<td>d.LF</td>
<td>Duration of the lean forward phase in the sit-to-stand transition</td>
<td>seconds</td>
</tr>
<tr>
<td>d.LT</td>
<td>Duration of the lift phase in the sit-to-stand transition</td>
<td>seconds</td>
</tr>
<tr>
<td>a.S2ST</td>
<td>Maximum change of the trunk angle in the lean forward phase</td>
<td>degrees</td>
</tr>
<tr>
<td>v.LF</td>
<td>Maximum angular velocity during the lean forward phase</td>
<td>degrees/s</td>
</tr>
<tr>
<td>v.LT</td>
<td>Maximum angular velocity during the lift up phase</td>
<td>degrees/s</td>
</tr>
<tr>
<td>d.ST2S</td>
<td>Duration of the stand-to-sit transition</td>
<td>seconds</td>
</tr>
<tr>
<td>d.PS</td>
<td>Duration of the prepare-to-sit phase in the stand-to-sit transition</td>
<td>seconds</td>
</tr>
<tr>
<td>d.SD</td>
<td>Duration of the sit-down phase in the stand-to-sit transition</td>
<td>Seconds</td>
</tr>
</tbody>
</table>
Parameter Extraction

To determine the duration of the entire TUG test and its separate phases, we need to detect and timestamp the following events:

- the beginning of the test \( T_{bTUG} \), which is also the beginning of the sit-to-stand transition \( T_{S2ST} \);
- the end of the sit-to-stand transition \( T_{eS2ST} \);
- the start of the stand-to-sit transition \( T_{ST2S} \); and
- the end of the stand-to-sit transition, which also corresponds to the end of the entire test \( T_{eTUG} \).

The first step in detecting the beginning of the sit-to-stand transition is to search for a change in the angular velocity around the frontal axis (the \( x \) component of the gyroscope). This change is caused by leaning forward as the individual prepares to stand up. The angular velocity has a distinct profile as illustrated in Figure 51, left. It starts from zero or near zero, increases to reach the maximum and drops to zero at the end of the lean forward phase. To determine the
beginning of the transition we first find the maximum angular velocity \(v_{LF}\) that is above a certain threshold and then search backward to find the beginning of the slope. The lean forward phase ends when the angular velocity drops back to zero \(T_{eLF}\). At this moment the maximum upper trunk angle, \(a_{S2ST}\), is reached (Figure 51, left). The time distance between the beginning of the transition and the end of the lean forward phase represents the duration of the lean forward phase, \(d_{LF} = T_{eLF} - T_{bS2ST}\). The second phase of the sit-to-stand transition is characterized by the negative angular velocity as the subject moves into the upright position. The angular velocity reaches the minimum, \(v_{LT}\), and then increases back to zero. The moment when it becomes positive is considered to be the end of the lift up phase and the end of the sit-to-stand transition. By time stamping this moment, we calculate the duration of the lift up phase, \(d_{LT}\), and the total duration of the sit-to-stand transition, \(d_{S2ST} = d_{LF} + d_{LT}\).

The stand-to-sit transition can also be divided into two separate phases, a prepare-to-sit, \(PS\), and a sit-down, \(SD\), phase. The angular velocity and the upper trunk angle profiles during the stand-to-sit transition are shown in Figure 51, right. The angular velocity increases to the maximum and then drops back to zero as the subject leans forward in preparation to sit down. The moment when the angular velocity drops to zero marks the end of the prepare-to-sit phase and the beginning of the sit-down phase. By time stamping these characteristic points we can determine the duration of the preparation phase, \(d_{PS} = T_{ePS} - T_{bST2S}\). In the sit-down phase the angular velocity is negative as the subject’s upper trunk moves back into the upright position. The moment it becomes the positive marks the end of the sit-down phase and the entire stand-to-sit transition. By time stamping this moment we can calculate the duration of the sit-down phase, \(d_{SD}\), and the entire stand-to-sit transition \((d_{ST2S} = d_{PS} + d_{SD})\).
5.2.3 Smartphone Application

sTUG smartphone application captures the signals from the smartphone sensors, processes the data to extract the parameters described in previous sections, and displays the parameters at the end of the TUG test. In addition it creates an iTUG test descriptor that includes time and date when the test is taken as well as all the parameters from Table 10. This descriptor is stored on the smartphone. It can optionally be uploaded to a medical server for long-term storage and analysis.

The user starts monitoring and processing by pressing a start button on the screen. Alternatively, the application and monitoring can be started automatically using an inexpensive Near Field Communication (NFC) tag. The application stops monitoring automatically once it detects the end of the stand-to-sit transition. Figure 52 shows a report generated by the application at the end of a TUG test.

sTUG is developed for Android operating systems and requires a smartphone with the accelerometer and gyroscope sensors running Android 2.3 or above. The application has been tested on a Nexus 4 smartphone, a Motorola RAZR M, and a RAZR HD.
5.2.4 Results, Discussion, and Conclusion

A preliminary testing of the sTUG application is performed on three individuals with diagnosed Parkinson’s disease and four healthy individuals. Each individual was asked to perform the TUG test in the shortest amount of time and the test is repeated three times. Table 11 shows the summarized results with all parameters reported by the sTUG application. Expectedly, the individuals with Parkinson’s disease needed more time to complete the total test ($d_{TUG}$) as well as the individual phases of the test. The healthy individuals had notably higher the maximum angular velocity during the lift up phase ($v_{LT}$). The duration of the stand-to-sit phase for healthy individuals was notably shorter.
### Table 11 iTUG Parameters for 3 Individuals with diagnosed Parkinson’s disease (S1-S3) and 4 healthy individuals (S4-S7).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>11.65±0.99</td>
<td>2.59±0.08</td>
<td>1.27±0.18</td>
<td>1.32±0.24</td>
<td>52.78±4.12</td>
<td>124.77±27.54</td>
<td>87.17±1.38</td>
<td>3.15±0.53</td>
<td>1.68±0.22</td>
<td>1.47±0.31</td>
</tr>
<tr>
<td>S2</td>
<td>11.45±0.62</td>
<td>2.13±0.44</td>
<td>1.14±0.30</td>
<td>1.27±0.19</td>
<td>55.54±10.89</td>
<td>141.23±19.24</td>
<td>88.45±21.67</td>
<td>2.28±0.03</td>
<td>1.23±0.15</td>
<td>1.05±0.17</td>
</tr>
<tr>
<td>S3</td>
<td>14.4±0.68</td>
<td>2.74±0.71</td>
<td>1.18±0.04</td>
<td>1.27±0.13</td>
<td>44.84±10.87</td>
<td>94.46±27.13</td>
<td>55.02±17.67</td>
<td>3.02±0.33</td>
<td>1.81±0.31</td>
<td>1.21±0.22</td>
</tr>
<tr>
<td>S4</td>
<td>7.03±0.24</td>
<td>2.65±0.27</td>
<td>1.13±0.28</td>
<td>1.15±0.96</td>
<td>56.82±15.96</td>
<td>115.2±36.29</td>
<td>100.0±6.75</td>
<td>1.51±0.04</td>
<td>0.6±0.03</td>
<td>0.9±0.07</td>
</tr>
<tr>
<td>S5</td>
<td>4.92±0.09</td>
<td>1.75±0.11</td>
<td>0.7±0.03</td>
<td>1.05±0.13</td>
<td>76.28±1.57</td>
<td>189.02±6.67</td>
<td>164.71±19.11</td>
<td>1.31±0.03</td>
<td>0.5±0.01</td>
<td>0.8±0.02</td>
</tr>
<tr>
<td>S6</td>
<td>5.15±0.11</td>
<td>1.61±0.12</td>
<td>0.73±0.10</td>
<td>0.88±0.03</td>
<td>71.72±4.09</td>
<td>207.07±11.85</td>
<td>221.18±12.68</td>
<td>1.65±0.07</td>
<td>0.81±0.03</td>
<td>0.84±0.06</td>
</tr>
<tr>
<td>S7</td>
<td>6.86±0.49</td>
<td>2.23±0.17</td>
<td>0.97±0.11</td>
<td>1.26±0.23</td>
<td>59.43±9.39</td>
<td>146.41±18.22</td>
<td>126.26±24.56</td>
<td>1.5±0.32</td>
<td>0.7±0.18</td>
<td>0.8±0.24</td>
</tr>
</tbody>
</table>

The testing showed promising results. By utilizing commodity smartphones we offer an affordable tool for instantaneous quantification of the iTUG tests. The test can be performed at home to assess the impact of drugs. Longer term analysis of the parameters may help in tracking progression of mobility and balance impairments.

5.3 **Real-time Monitoring of Activity of Wheelchair Users**

Public health groups have been advocating for an increase in physical activity levels as a way to lower the risk of inactivity-related diseases, such as cardiovascular disease, insulin resistance, hyperglycemia, and type 2 diabetes. Physically inactive individuals are almost twice as likely to develop coronary heart disease when compared to those who exercise regularly. Recent estimates suggest that the impact of physical inactivity on mortality risk is approaching tobacco as one of the leading causes of death in the able-bodied population [50].

People with limited ambulatory skills who use wheelchairs for mobility are especially at high-risk for all inactivity-related diseases. For example, it has been reported that a person with a spinal cord injury (SCI) has a significantly greater risk of mortality from coronary heart disease (225%) than an able-bodied person [50]. According to a 2005 U.S. Census Bureau’s Survey, over 3.3 million Americans use some type of wheelchairs for mobility and with aging population this number is likely to continue to grow.
Many organizations developed specific physical activity recommendations for disabled adults. The U.S. Department of Health and Human Services recommends that disabled adults should accumulate 150 minutes per week of moderate-intensity and 75 minutes per week of vigorous-intensity aerobic activity [133]. To quantify compliance with these recommendations, a method that can accurately estimate physical activity for this group of individuals is needed.

To meet the challenge and provide an affordable, reliable, and easy to use solution for monitoring the physical activity of users who rely on wheelchairs for mobility we developed a smart wheelchair – a common wheelchair instrumented only with a smartphone that is used to track a user’s physical activity. We utilized the smartphone’s built-in sensors such as a magnetic sensor for monitoring wheelchair speed and distance traveled, an accelerometer for monitoring smartphone’s orientation and wheelchair inclination, and a proximity sensor to determine whether the wheelchair is hand-propelled or pushed. In addition, we employ a wearable chest belt to monitor and record the user’s heart activity and energy expenditure. A smartphone application called mWheelness collects the data from the sensors and performs periodic uploads to the mHealth server.

5.3.1 Related Work

There have been a number of studies focusing on activity monitoring in the able-bodied population that rely on step counters and other accelerometer-based systems or on surveys [134], [135]. However, despite having a variety of different methods for the measurement of physical activity, each has its own limitations for manual wheelchair users. For example, pedometers and other step activity monitors will not apply to individuals who are unable to walk. Several studies focus on suitability of accelerometers for assessment of physical activity of individuals in wheelchairs [136], [137], [138], however they often require expensive and sophisticated wheelchair monitors and/or user instrumentation with wearable electronics that may impede normal daily activities. Alternatively, surveys are commonly used for the elderly and even
individuals with disabilities. However, there are inherent problems with subjectivity and inaccuracy from recall and interpretation. This is of special concern for individuals who use wheelchairs because they often spend the majority of time in low-level activities.

5.3.2 Methods

The system can record, log, display, and communicate information about the user’s physical and heart activity during normal daily activities or exercise sessions. For monitoring the user’s physical activity (e.g., speed and distance travelled) we rely solely on smartphone’s built-in sensors, including a magnetic sensor, an accelerometer, and a proximity sensor, while for monitoring heart activity we use a wearable chest belt.

Wheelchair instrumentation

Figure 53 illustrates the proposed wheelchair instrumentation with a smartphone. The smartphone is placed in a holder on a side of the wheelchair. A small magnet is attached on a wheel of the wheelchair. The smartphone’s magnetic sensor is sensing the Earth magnetic field and is normally used to aid navigation by determining the Earth’s magnetic poles and the smartphone’s orientation. The magnetic sensor senses the x, y, and z components of the magnetic field as illustrated in Figure 53 (the actual orientation may vary across different smartphones). By placing a small magnet on the wheel, we induce a change in the magnetic field sensed by the magnetic sensor of the smartphone when the magnet moves over the smartphone. This change produces a characteristic signature in the magnetic field signals that can be sensed, recorded, and processed on the smartphone. By processing the magnetic field signals we can detect and timestamp an event - when the magnet moves right over the smartphone which corresponds to one revolution of the wheelchair’s wheel.

A smartphone’s accelerometer measures proper acceleration and is typically used to keep the screen upright regardless of the smartphone orientation. In our setup we process the x, y, and z acceleration components to determine smartphone’s orientation, i.e., whether it is placed in the
wheelchair holder or not. Activity recording is enabled only when the smartphone is properly mounted on the wheelchair. In addition, the accelerometer data is used to determine slope of the wheelchair which can further be used to determine vertical gain and loss during exercise.

![Smartphone instrumentation of a wheelchair](image)

*Figure 53 Smartphone instrumentation of a wheelchair*

A smartphone’s proximity sensor is typically used to determine when the smartphone is brought up to the user’s ear and usually acts as a binary sensor. In our deployment, the smartphone’s proximity sensor is used to determine whether the user hand-propels the wheelchair or it is pushed. This information can be used to further qualify the user’s activity.

**Measuring speed and distance**

Figure 54 shows the raw magnetic field components recorded by our custom application running on a smartphone mounted on a wheelchair as shown in Figure 53. The wheelchair is placed on a treadmill that moves at the constant speed of 1 mile per hour (mph) (Figure 54a), 2 mph (Figure 54b), and 4 mph (Figure 54c). Whereas all three axes of the magnetic field (x, y, and z) show a distinct signature when the magnet moves over the smartphone, the changes in the z component are the most distinct. Consequently, our algorithm for detection of a wheel revolution
focuses on this component of the magnetic field. The signal is such that even a simple visual inspection may be used to confirm the feasibility of the proposed approach -- changes in the magnetic field induced by the magnet can indeed be used to determine the number of wheel revolutions. For example, we can identify 5 peaks in the $z$ component of the magnetic field during a window of 20 seconds (Figure 54a). This corresponds to 5 revolutions of a 24” diameter wheel, which in turn corresponds to ~1 mph speed. Similarly, we can identify 19 peaks when the treadmill belt moves at 4 mph.

Figure 54 Smartphone recordings of the $x$, $y$, and $z$ components of the magnetic field for different wheelchair speeds: (a) 1 mph; (b) 2 mph; and (c) 4 mph.

Whereas the treadmill experiment confirms feasibility of the proposed approach, it presents somewhat idealized conditions. The wheelchair is fixed on the treadmill and its position relative to the Earth’s magnetic field remains unchanged during the experiment. In real-world conditions the wheelchair will constantly change its position relatively to the Earth’s magnetic field. This will result in changing the base values of the magnetic field observed by the smartphone’s magnetic sensor.

Figure 55a shows the $z$ component of the magnetic field recorded during a freewheeling experiment when a user propels himself in a wheelchair moving through a square-shaped
hallway, making a full round trip of approximately 100 meters in length. We can observe a slow-changing DC component of the magnetic field due to changes in the smartphone position relative to the Earth’s magnetic field. Fortunately, the spikes induced by the proximity of the magnet on the wheel when the magnet moves over the smartphone can still be easily detected.

![Figure 55](image)

*Figure 55 From signals to information: (a) z component of the magnetic field; (b) processed z component; (c) calculated speed; and (d) proximity sensor data.*

The next step is to transform the original signal representing the z component of the magnetic field into a signal that allows for efficient detection of peaks. By detecting and time-stamping peaks in the transformed signal, we can precisely derive the wheelchair speed and distance traveled. In developing the signal processing algorithm our goal is to achieve accuracy and reliability at minimum cost in computation time and energy consumed.

The transformation should remove a slow changing component caused by changes in the smartphone orientation relative to the Earth’s magnetic field and amplify the component of the signal caused by the proximity of the magnet on the wheel. An incoming sample, \( z(i) \), is stored in a sliding buffer that keeps a window of 40 most recent samples -- this corresponds to 1 second in time with the sampling frequency of 40 Hz. Note: in general the window size is a function of the
sampling frequency. The processed signal, \( p_z \), is calculated as shown in (5-2). Figure 55b shows the transformed signal.

\[
pz(i) = \left| z(i) - \frac{1}{40} \sum_{j=-19}^{j=20} z(i + j) \right|
\]  

(5-2)

The second step is to detect peaks in the processed signal \( p_z \). Our algorithm for peak detection involves a state machine with three states, Pre-Peak, Expect-Peak, and Post-Peak. Pre-Peak is maintained as long as the processed signal samples, \( p_z(i) \), are below a certain threshold. When a sample is above the threshold we enter the Expect-Peak state. The algorithm exits the Expect-Peak state once the processed samples are below the threshold. The threshold is set at 13 µT. In this state the samples are buffered and searched for the maximum. The sample number of the maximum is used to precisely timestamp the moment when the peak occurred. To avoid false positives, Post-Peak is entered. The state machine remains in this state for a predefined period of time. The duration of the Post-Peak state is determined by the maximum speed we would like to detect. For example, at the speed of 10 mph, which is twice as the realistic maximum speed of 5 mph, the time distance between two peaks is 0.428 seconds for a wheelchair with 24” wheel. This means that in normal operation no two valid peaks can possibly be detected in a period shorter than 0.428 seconds.

Figure 55b shows the transformed signal with blue circles identifying peaks detected by the proposed algorithm. The visual inspection indicates that algorithm achieves 100% accuracy. The algorithm has been extensively tested in both treadmill experiments and in hand-propelled wheelchair experiments.

Measuring speed and distance travelled can easily be derived from the sequence of time-stamped peaks. Let us assume a sequence of peak timestamps \( t_i, t_l, t_{l+1} \). The speed at time \( t_i \), \( v(t_i) \), is calculated as shown in(5-3), where \( d \) is the diameter of the wheel. If no peak is detected during a predefined time window (e.g., 8 seconds), the speed is reset to zero. Figure 55c shows the
calculated speed over time for the free-wheeling experiment. The distance travelled, $D$, is maintained by simply counting the number of detected revolutions, $N$, and can be calculated using (5-4).

$$v(t_i) = \frac{\pi d}{t_i - t_{i-1}} \quad (5-3)$$

$$D = \pi d N \quad (5-4)$$

In designing the peak detection algorithm and speed estimation we need to consider the sampling frequency of the smartphone’s magnetic sensor. Higher sampling frequency provides better resolution of timestamps and consequently more accurate speed estimation. Lower sampling frequency may reduce the compute complexity and storage requirements for the peak detection as well as the power consumption of the smartphone. In determining the sampling frequency we start from an application requirement - what is required accuracy for speed estimation. For example, if we want to achieve accuracy of $\Delta v = 0.1$ mph, the minimum required sampling frequency, $F_s$, for a given speed $v$ can be determined using (5-5). Thus, to distinguish between 0.9 mph and 1 mph, the minimum sampling frequency is 2.1 Hz (assuming a 24” wheel). If we want to distinguish between 3.9 and 4.0 mph, the minimum required sampling frequency is 36.4 Hz. Finally, if we want to distinguish between 5.9 mph and 6.0 mph, the minimum sampling frequency is 82.6 Hz. For all practical applications where wheelchair speed is rarely to exceed 4 mph, the sampling frequency of 40 Hz is adequate.

$$F_s = \frac{1}{\frac{\pi d}{v - \Delta v} - \frac{\pi d}{v}} \quad (5-5)$$

**Detection of self-propelled vs. assisted motion**

The smartphone’s proximity sensor can be utilized to determine whether the wheelchair is propelled by the user or possibly pushed by another person. A modern smartphone’s proximity sensor typically reports only two states – an object is closer or farther than triggering distance of the sensor. Depending on the type of the sensor built in the smartphone, the triggering distance is
typically 3-10 cm. The proposed placement of the smartphone on the wheelchair makes it possible to detect hand propulsion -- when the user places his/her hand on the hand rim of the wheel he/she will likely trigger the proximity sensor. When the wheelchair is pushed by another person, the user’s hand is not likely to interfere with the smartphone’s proximity sensor.

Our approach is to simply maintain a status of the proximity sensor: if it reported a change in a certain time window, we consider the wheelchair to be hand-propelled by the user. Figure 55d shows the raw values reported by the proximity sensor of a Motorola RAZR M smartphone during the free-wheeling experiment. The proximity sensor reports two distances, 3 cm and 100 cm. Frequent changes of the reported distance indicate hand propulsion during the entire experiment.

Whereas the proposed algorithm works well in our initial studies, it has its limitations – the user’s hand may not always be close enough to trigger the proximity sensor and detection strongly depends on triggering distance of the proximity sensor. An alternative approach, albeit costlier in terms of processing and storage requirements, is to process still images or video captured by the smartphone camera in order to detect whether the user hand propels the wheelchair.

**Tilt and Incline Calculation**

Tracking of physical activity is conditioned by the smartphone being in the upright position in the wheelchair holder. This can be detected by calculating the smartphone tilt as shown in Figure 56a. The angle $\phi$ can be estimated as in (5-6). The tilt of the wheelchair in normal operation is unlikely to exceed a certain range (e.g., -10 to +10 degrees) and we use this property to determine whether the smartphone is in the holder or not. If this angle exceeds a certain predefined range, we consider the smartphone to be taken by the user, and the processing of the magnetometer signals for detection of wheel revolutions is suspended.
In a similar fashion we calculate the smartphone slope as shown in Figure 56b. Whereas the wheelchair propulsion will have an impact on the $x$ and $z$ components of the acceleration, $A_x$ and $A_z$ respectively, the $x$ and $y$ components of the acceleration, $A_x$ and $A_y$, will still be dominated by the Earth’s gravity. By filtering the calculated angle $\theta$ (5-6), we can extract the slope of the wheelchair and thus use this information to determine the vertical gain and drop during an exercise session. An alternative approach to eliminate the impact of forces due to propelling is to consider only acceleration samples when the square root of the sum of squared acceleration components is very close to the Earth gravity (9.81 m/s$^2$).

The described algorithms for determining the smartphone’s tilt and incline assume a perfect alignment of the wheelchair holder -- angles $\phi$ and $\theta$ are both equal to 0 degrees when the wheelchair is on flat surface as shown in Figure 56. To accommodate for imperfect holder placement, we go through a calibration process where we calculate offsets $\phi_o$ and $\theta_o$, and the measured angles are compensated for these offsets.

An alternative approach to determining that the smartphone is in the wheelchair holder is to place an inexpensive NFC tag in the holder. The NFC capable smartphones can utilize this tag.
to detect when the smartphone is placed in the holder. In addition, this event can be used to automatically start the smartphone application.

**Monitoring Heart Activity and Estimating Energy Expenditure**

To monitor the user’s heart activity we employ a commercially available Zephyr HXM heart monitor [40] that can be paired with the smartphone over Bluetooth. The monitor is mounted on a chest belt with textile electrodes. It sends a message every second containing the average heart rate and time stamps of the last 15 heart beats. mWheelness estimates energy expenditure in calories, $\Delta C$, using (5-7) for men and (5-8) for women as proposed by Keytel et. al. [139]. It is a function of the heart rate (HR), the user’s weight in kilograms (W), and the age in years (A). We update the total energy every second ($\Delta T=1$ sec).

$$\Delta C = \frac{(-55.0969 + 0.6309 \cdot HR + 0.1988 \cdot W + 0.2017 \cdot A) \cdot \Delta T}{4.184}$$  \hspace{1cm} (5-7)

$$\Delta C = \frac{(-20.4022 + 0.4472 \cdot HR - 0.1263 \cdot W + 0.074 \cdot A) \cdot \Delta T}{4.184}$$  \hspace{1cm} (5-8)

**5.3.3 Smartphone Application**

mWheelness smartphone application implements the algorithms described in the previous section to support activity monitoring and logging. In addition, it supports monitoring and logging of heart activity. The recorded logs of registered users can optionally be uploaded to mHealth Databases.
Figure 57. mWheelness Android application screens

Figure 57 shows characteristic screens of mWheelness. A configuration screen shown on the left allows a user to enter wheel diameter, as well as the sampling rate, and the magnetic sensor detection threshold. In addition, the screen allows for configuring what type of information will be logged into a file on the smartphone and/or uploaded to the databases. Although gender, weight, age will be populated from mHealth Database when the user sign in the application, configuration screen allows entering these information in case they have to be changed. It also guides the user through a calibration process providing acoustic clues to start calibration, place the smartphone in the holder, and remove it after the calibration is performed.

The user starts recording physical activity and heart activity by pressing the Start/Stop Recording button – please note that processing of the signals from the magnetic sensor will not start before the smartphone is in the upright position. During an exercise session mWheelness displays current inclination, speed, and distance travelled. In addition, it displays information about heart activity.
5.3.4 Results, Discussion, and Conclusion

The mWheelness application is tested on several Android smartphones in controlled and free-living conditions. The controlled experiments are conducted on a treadmill while varying speed and inclination. The distance traveled and inclination reported by the application are compared against the corresponding parameters reported by the treadmill. mWheelness can run in a special mode that has additional support for logging all raw samples from the smartphone sensors. This allows us to verify inner workings of our algorithms offline – the raw data are processed in Matlab to generate events that are compared against the events logged by the mWheelness.

Free-wheeling experiments involved five able-bodied individuals hand propelling the wheelchair on a rectangular-shaped course of 201 meters in length. As test platforms we used Motorola RAZR M and HTC One X. Each individual completed the course two times, once in each direction. The average distance and standard deviation reported by mWheelness is 198.8±3.1 meters for Motorola RAZR M and 200±3 meters for HTC One X. This is an excellent result if we know that the maximum precision we can achieve is within two circumferences of the wheel from the course length (3.8 meters in our case). The RAZR M’s proximity sensor with triggering distance of 3 cm occasionally misses hand movements during hand propelling for some users, whereas the HTC One X’s proximity sensor with triggering distance of 9 cm performed flawlessly.
Figure 58 Heart Rate, Speed, and Hand Propelling during a freewheeling experiment

Figure 58 shows user’s heart activity, wheelchair speed, and hand propelling flag recorded by mWheelness during an experiment. The wheelchair is pushed for the first 75 seconds, and hand propelled by the user afterwards. We can observe a steep increase in heart rate once the user started hand propelling.

To estimate maximum operating time of the mWheelness application with a single battery charge, we conducted several experiments on Motorola RAZR M. The operating time when only physical activity is logged (with WLAN, Bluetooth, and mobile data turned off) is slightly over 20 hours. The operating time when both physical and heart activity are logged (with WLAN and mobile data turned off, Bluetooth is on) is slightly over 18 hours. The entire system is currently undergoing testing in a local rehabilitation hospital in both supervised and free-living conditions.

5.4 Real-time Monitoring of Occupational Stress of Nurses

The Bureau of Labor Statistics reports that all occupations are exposed to stress, but nearly 25% of stress is attributed to occupations involving healthcare workers [140]. Situations reported to produce stress for hospital workers, in particular nurses, are the inadequate staffing of
nurses for the number and acuity of patients, schedules (length of shift, recovery time between days worked, and night shifts), lack of recovery time while on shift (being on feet all day), interruptions, inefficient hospital systems, and the absence of unlicensed personnel to assist nurses with physically demanding tasks [141], [142]. Lindberg et al. [143] found that registered nurses on average complete over 160 tasks during 8 hours (2.5 minutes per task). During their work, nurses are constantly interrupted by telephone calls, beeps, and alarms; Hall et al. [144] observed over 13,000 disruptions in the work of 360 nurses in a 2-week period. Solomon et al. [145] found that interruptions comprised over 60% of the factors contributing to workplace stress in hospitals. This unrelenting demand on nurses causes stress and strain. Salmond and Ropis [146] found that stress caused by frequent interruptions impacts the general health and well-being of nurses. Psychological stress is also experienced by nurses. These include fatigue from dealing with difficult patients, families, and other healthcare providers, stress from caring for ill or dying patients, and lack of autonomy in care delivery decisions or lack of input into unit/hospital decisions [147].

From an organizational point of view, occupational stress of nurses is an important factor because of its relationship to employee productivity and performance. Occupational stress can lead to absenteeism, burnout, and turnover [148]. These consequences of chronic stress are not only harmful to healthcare organizations where nurses work, but they affect patients. High stress in the hospital work environment has a strong correlation to increased numbers of patient care errors according to Dugan et al. [149]. When a sufficient number of stressed nurses work together, the satisfaction of patients on that hospital unit suffers [150].

*Allostatic load* represents the long-term physiological consequences of changes in neural or neuroendocrine activity caused by stress. McEwen and Stellar introduced the term *allostatic load* to represent ‘the wear and tear on the body’ caused by exposure to repeated or chronic stress [151]. Individuals marshal physiological responses to a short, stressful situation that can be seen as an increase of stress hormones (cortisol) and heart rate and a decrease of heart rate variability.
Those changes are essential for adaptation, maintenance of homeostasis, and survival. After the stressful situation passes, the individual's heart rate will typically return to baseline relatively fast; however, the level of stress hormones remains elevated for 15-45 minutes [152]. Under multiple stressful situations or unresolved stress, the heart rate and other vital parameters may fail to return to baseline [151]. In addition to repeated stress and failure to habituate, allostatic load may be caused by failure to turn off stress responses in a timely manner or inadequate physiological response that leads to compensatory hyperactivity of other mediators. Even though allostatic load is conceptualized as a long-term dysregulation of neuroendocrine activity, its precursors are the moment-by-moment stressful situations found in everyday life. Therefore, it is important to detect early indications of allostatic load to prevent its profound negative effects on a workers’ physical and emotional well-being.

Allostatic load can be measured as imbalances in autonomic nervous system, central nervous system, neuroendocrine, and immune system activity. Many different physiological measurements can be used to detect allostatic load in workers [153]. The level of stress hormones can be measured reliably from serum, salivary, and urine samples [151], [152] Heart rate variability has been used as a measurement of the autonomic nervous system's response to stress [154], [155], [156] including job stress [157]. Blood pressure and blood pressure variability have also been studied as a correlate of general stress [158], [159] and job stress [160].

The purpose of our study [161], [162] was to examine the physiological responses of registered nurses during nursing care on a high-fidelity patient simulator, and to identify the best physiological parameters suitable for detecting occupational stress during simulated nursing activities.

5.4.1 Related Work

In order to detect and assess stress, many researchers approached inducing stress in the participants of the study. For this purpose they used many different methods, from mental
arithmetic [163],[164], memory tests [165],[166], and puzzles [167], audio and movie stimulants [168] to electric shocks [169]. However, we chose to simulate a hospital work environment as the means of inducing stress in participants. This approach allows us to get physiological responses similar to the responses in real environment.

5.4.2 Methods and Experimental Setup

We used a control room with a one-way mirror and a private patient room equipped with high fidelity simulator, functional headwall, state of the art physiological monitors, and audio and video monitoring systems. From the control room, we ran the simulated nursing activity, and using our custom software synchronously recorded the activity of the participant from a video camera and their physiological signals from a Zephyr BioHarness, placed as chest belt. All records of the physiological signals are automatically uploaded to the research server. The records contain interbeat intervals, heart rate, breathing rate, and acceleration on chest belt. A nursing student during a training session on the high fidelity patient simulator in the Real-time Physiological Monitoring Lab is shown on the Figure 59.

*Figure 59 Real-Time Physiological Monitoring Lab*
5.4.3 Participants and Simulation protocol

We used a non-probability sampling method to obtain research participants from students enrolled in either the Bachelor of Science in Nursing program or the Master of Science in Nursing program. Only registered nurses could participate in the study due to the type of simulation that was used. We excluded participants on the basis of three criteria: previous diagnosis of high blood pressure or irregular heart rhythm, latex allergy, or age less than 19 years. During online registration, we collected demographic data from the volunteers. A total of 14 participants were recorded: 12 female and 2 male subjects. The mean age of participants was 32.8 years old, with the youngest participant being 23 and the oldest participant being 46 years old.

We developed a simulation that required subjects to care for a patient with a tracheostomy used for breathing. We used a computer program to run the simulation in exactly the same order for each subject. The simulation protocol ran for 30 minutes for every subject and included the following activities:

- Measurement of subject’s heart rate, heart rate variability, and respiratory rate at rest for 5 minutes
- Reading “change of shift report” about patient’s status
- Assessment of the patient by the subject
- Disturbance #1: Telephone call to create an interruption during assessment
- Documentation of assessment findings by the subject
- Disturbance #2: Telephone interruption during documentation – request for pain medication
- Sudden respiratory distress of patient (coughing, difficulty breathing, low oxygen saturation, elevated heart rate and blood pressure) requiring urgent intervention by the subject
- Disturbance #3: Family member apprehension and questioning of subject’s actions
- End of simulation when subject pre-oxygenates and suctions patient to relieve mucous plug
- Five minute measurement of physiological parameters at rest.

### 5.4.4 Signal Processing

Typical parameters of Heart Rate Variability (HRV) are calculated from RR intervals on 60 second window with 30 second overlap. A small window size is selected to support future real-time monitoring and warning system implementation. Analyzed parameters include Root Mean Square of Successive RR Differences (RMSSD), spectral power in Low Frequency (LF) and High Frequency (HF) bands using autoregressive (AR) and Fast Fourier Transform (FFT) methods [41].

### 5.4.5 Results

A summary of the selected physiological parameters for all subjects is presented in Table 12. The results are described in the paragraphs below.

**Table 12 Summary of the selected physiological parameters for all subjects**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Before Simulation</th>
<th>During Simulation</th>
<th>After Simulation</th>
<th>Δ for Call #1</th>
<th>Δ for Call #2</th>
<th>Δ for Patient in Crisis</th>
</tr>
</thead>
<tbody>
<tr>
<td>HR [bpm]</td>
<td>82.76±12.29</td>
<td>96.61±14.01</td>
<td>84.85±14.86</td>
<td>9±7.99</td>
<td>8.63±6</td>
<td>17.38±10.39</td>
</tr>
<tr>
<td>RMSSD [ms]</td>
<td>38.86±21.2</td>
<td>29.81±13.6</td>
<td>37.65±18.65</td>
<td>-2.84±9.76</td>
<td>3.64±12.18</td>
<td>-7.06±20.98</td>
</tr>
<tr>
<td>AR LF/HF</td>
<td>2.16±1.77</td>
<td>4.68±2.69</td>
<td>2.4±1.48</td>
<td>-0.97±3.47</td>
<td>0.14±4.74</td>
<td>1.66±6.02</td>
</tr>
<tr>
<td>FFT LF/HF</td>
<td>2.36±2.62</td>
<td>5.18±3.04</td>
<td>2.42±1.41</td>
<td>-0.64±5.17</td>
<td>-2.15±9.9</td>
<td>1.34±7.86</td>
</tr>
<tr>
<td>Breathing rate [bpm]</td>
<td>16.88±2.24</td>
<td>17.73±3.72</td>
<td>16.79±4.34</td>
<td>0.9±3.05</td>
<td>-0.75±5.11</td>
<td>0.69±3.53</td>
</tr>
<tr>
<td>Acceleration Magnitude [g]</td>
<td>0.04±0.02</td>
<td>0.1±0.01</td>
<td>0.04±0.02</td>
<td>0.02±0.05</td>
<td>0.01±0.04</td>
<td>0.03±0.06</td>
</tr>
</tbody>
</table>

Three particular events from the aforementioned simulation protocol were expected to be stressful for the subjects: two interruptions by telephone call (Disturbances #1 and #2) and sudden respiratory distress of patient. We used repeated measures analysis of variance to evaluate changes in physiological data before, during, and after training session, as well as instant changes caused by three potentially stressful events.
The training session created significant strain on subjects. On average, heart rate increased 16.7%, from 82.8 to 96.6 bpm, falling to a slightly increased level after the training session (84.9 bpm). All subjects had increased heart rate even before the training session that can be attributed to anxiety in anticipation of the session. The prolonged effects of the stress are visible in higher heart rate and lower heart rate variability (e.g., Root Mean Square of Successive RR Differences decreased from 38.9 ms to 37.7 ms). The breathing rate increased also during the simulation from 16.9 to 17.7 breaths/min.

The ratio of the low-frequency and high-frequency components of the RR spectrum (LF/HF) significantly changed during the simulation from 2.16 to 4.68 and followed the same pattern as heart rate – increased during the simulation and slowly return to normal after the session 2.4.

The sudden respiratory distress of simulated patient created significant physiological responses in subjects. The average heart rate increased 17.4 bpm at the onset of the event. Disturbances also significantly influenced physiological parameters by increasing demands in an already stressful situation. The first telephone call increased the heart rate on average 9 bpm, while the second call increased the heart rate 8.6 bpm.

5.4.6 Discussion and Conclusion

The described study examined the physiological responses of nurses during care on a high-fidelity patient simulator to develop research methodology and identify the best physiological parameters suitable for real-time assessment of allostatic load during work.

Our results indicate that physiological parameters extracted from wearable monitors, such as heart rate, heart rate variability (RMSSD, LF/HF), breathing rate, and physical activity provided a reliable indication of stress-induced physiological changes. Our findings are consistent with previous research showing that heart rate, heart rate variability, and breathing rate are
measures of the short-term cardiovascular responses to stressful situations such as patient in crisis [155], [157], [170].

The important contribution of this study is assessment of physiological changes caused by work-place related disturbances, such as telephone calls. The disturbances created significant short-term cardiovascular responses that contribute to overall allostatic load. Lackner and Goswanmi reported similar changes when researchers gave instructions during a math task as the stress source [171]. Interruptions are particularly important if we consider the number of work related interruptions. In a typical nursing work environment, there are approximately 30 interruptions per shift [172], with numerous interruptions during drug administration [173].

Continuous real-time monitoring also provides quantification of allostatic load using the dynamics of physiological changes with a ‘new normal’ of heart rate and heart rate variability after the stress. Because wearable physiological monitoring can be used during work and leisure, we can track long term changes of the selected parameters to examine allostatic load.

5.4.7 Future Work

Future work includes long-term wearable monitoring (weeks instead of hours), automatically triggered self-reports, and the development of personalized assessment of the allostatic load. Monitoring for extended periods will provide data for long-term trends and model development. Self-reported questionnaires triggered by the peak allostatic load on smartphones will facilitate collection of data regarding the perceived level and causes of stress in specific work settings.

The real-time assessment of allostatic load can be used to provide real-time alerts for nurses to engage in stress reducing activities. Objective assessment of the allostatic load can be also used by the hospital administrators to understand job stress in their own settings and adjust nurse staffing or patient assignments to improve quality of patient care and quality of life for nurses.
Chapter Six

Conclusion

The existing health care system is in imminent crisis caused by current economic, social, and demographic trends. Significant increase of healthcare costs (up to 4.5 trillion by 2020) and increased life expectancy in combination with retiring Baby Boomers further contributes to the healthcare crisis. Furthermore, the existing health care system is centralized and focused on reacting to illness. This cause several issues: (i) centralized solutions are hard to scale, (ii) illnesses are typically discovered late, and (iii) late discovery of illness increases costs and significantly reduces chances for recovery.

Technology advances enabled development of new health care systems based on wearable physiological monitors that promise to shift the focus on proactive wellness management and ubiquitous health monitoring. Wearable physiological monitoring has the potential to distribute healthcare services from hospitals and medical centers to individuals and their homes. Moreover, continuous wearable health monitoring has the potential to engage users, improve wellness management, prevent disease by early detection, and assist rehabilitation and treatments.

This dissertation considers computing infrastructure (both hardware and software aspects) to support mobile health and wellness monitoring applications as well as services to support further research in the area of wearable health monitoring. Specifically, we focus on a multi-tiered organization of the mHealth infrastructure that includes wearable sensor nodes at Tier 1, personal applications running on mobile computing platforms at Tier 2, and server and...
services at Tier 3. We consider several important aspects of the mHealth systems including time-synchronization to support distributed real-time monitoring and energy-efficiency. Finally, the dissertation makes the case and introduces several original mHealth applications that are fully implemented and tested. These include monitoring and capturing dynamic heart response to posture transitions, quantification of timed-up-and-go tests, monitoring of physical activity of wheelchair users, and monitoring of occupational stress of nurses.

The main contributions of this dissertation are as follows:

- The development of framework for distributed wearable physiological monitoring,
- An implementation of the mHealth infrastructure at the University of Alabama in Huntsville,
- An evaluation of design space for synchronized distributed wearable physiological monitoring,
- The design and implementation of synchronized distributed monitoring that allows global monitoring across the planet with sub-second time synchronization,
- An evaluation of the design space for improved energy efficiency of wearable physiological monitoring systems,
- The design and implementation of the automated environment for dynamic power profiling optimized for mobile devices,
- An implementation of optimized application-specific processor architecture for wearable sensors,
- The design and implementation of an environment for user guidance and synchronous physiological data and video recording during structured experiments,
- Analysis and implementation of novel approaches in utilization of the smartphone’s built-in sensors for physiological monitoring, and
• Introduction and original implementation of several wearable monitoring applications utilizing mHealth infrastructure.

Our evaluations of the design space for improvement of energy efficiency shows several important findings:

• Simple customization of processor on the physiological sensors can reduce energy consumption of its signal processing tasks up to 5 times compared to software implementation on the same processor without customization.

• Energy consumption of the smartphone can be reduced for more than 6 times during data transfer between smartphone and server by optimal selection of the working frequency of the smartphone’s CPU and type and level of compression.

• Energy consumption of the CPU on the server can be reduced more than 2 times by optimal selection of the working frequency of the CPU and number of threads for data processing.

The mHealth infrastructure described in this dissertation enables further research in the area of wearable physiological monitoring, including but not limiting to:

• Comprehensive testing of the mHealth infrastructure in real life deployments,

• The development of software tools for analysis of collected physiological data to quantify overall physiological state and to generate personalized recommendations that may lead to improvement of overall health status, and

• The development of new and novel mHealth personal applications.
References


