ARTIFICIAL INTELLIGENCE-ENABLED EDGE-CENTRIC SOLUTION FOR AUTOMATED ASSESSMENT OF SLEEP USING WEARABLES IN SMART HEALTH

Md Juber Rahman

ARTIFICIAL INTELLIGENCE-ENABLED EDGE-CENTRIC SOLUTION FOR AUTOMATED ASSESSMENT OF SLEEP USING WEARABLES IN SMART HEALTH

by

Md Juber Rahman

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major: Electrical and Computer Engineering

The University of Memphis

August 2021

DEDICATION

This dissertation is dedicated to my parents, who motivated me to work hard and dream big.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my Ph.D. supervisors Dr. Bashir I. Morshed and Dr. Chrysanthe Preza. Dr. Morshed recruited me for a National Science Foundation (NSF) project, funded my Ph.D. study, and mentored and guided me throughout my entire journey. Having the privilege of working with him has been a great learning experience for me as I have learned to approach a problem as an opportunity and take systematic and logical steps in developing a solution. Words really cannot describe how much grateful I am to him for his patience in reviewing many of my inferior drafts, encouragement for working hard, and invaluable suggestions and guidance. Thank you Dr. Morshed for your wholehearted supervision throughout the progress of this work.

I would like to thank Dr. Preza who is co-supervising me since Dr. Morshed left the University of Memphis. That transition period was really scary for me, and I am deeply indebted to Dr. Preza for extending her unwavering support. Special thanks to the members of my committee: Dr. Mohd Hasan Ali, Dr. Madhusudhanan Balasubramanian, and Dr. Frank Andrasik for their invaluable advice and feedback on this research. I would also like to thank Dr. Ruhi Mahajan and Dr. Brook E. Harmon who were always willing to guide me in this multi-disciplinary research and have vastly enriched my knowledge in this domain. I acknowledge the funding provided by NSF, Institute of Intelligent Systems, and Department of Electrical and Computer Engineering at the University of Memphis. Also, my internship with Samsung Research America was very helpful. Thanks to Dr. Jilong Kuang.

I also want to express my heartfelt gratitude to my lab-mates Dr. Ankita Mohapatra, Dr. Saleha Khatun, Dr. Babak Noroozi, Dr. Sabbir Bin Zaman, Sharmin Afroz, Haritha Gollakota, Tasnuba Siddiqui, Moriom R. Momota, and Mahfuzur Rahman, who were always happy to help and made the workplace a delight to work in. A special thanks to my beloved wife for her great support, my siblings, family, friends, and well-wishers whose encouragement and support helped me in sustaining this far. Thanks to my children, for their patience. I received great support from my teachers during all phases of my education. Especially, I would like to thank my favorite teachers at Bangladesh University of Engineering & Technology (BUET) Dr. Mohammad Ali Chowdhury (may the Almighty grant him the highest abode in paradise), Dr. M M Shahidul

Hassan, and Dr. Abdul Hakim. They helped me with numerous recommendations. Also, I would like to thank my past colleagues at Power Grid Co. of Bangladesh Md Afsarul Haque Nazmee, B.M. Mizanul Hassan, and Dr. Farid Ahmed P.Eng. for their support, and recommendations. Special gratitude to my high school teacher Sujit Bhattacharjee and my uncle in law Dr. Rafique Uddin.

ABSTRACT

Rahman, Md Juber, Ph.D. The University of Memphis. August 2021. Artificial Intelligence Enabled Edge-Centric Solution for Automated Assesment of Sleep Using Wearables in Smart Health. Major Professor: Dr. Bashir I. Morshed and Dr. Chrysanthe Preza.

Artificial intelligence-enabled applications on edge devices have the potential to revolutionize disease detection and monitoring. Smartphones, with increased processing speed, storage capacity, and integrated sensors, hold tremendous promise in physiological sensing, monitoring, and development of Smart and Connected Communities (SCC). The major challenges are development of user-friendly low-cost sensing of physiological data, seamless data transfer while maintaining privacy, and data processing and inference at the edge while keeping the system performance at an expected level. Sleep Health evaluation is a prominent area where the advantages of edge-centric sHealth solutions can be leveraged. Rapid changes in socio-economic structure are impacting sleep health globally. Sleep apnea, insomnia, sleep deprivation, etc. are growing problems and impacting the health-related quality of life. To minimize the adverse health consequences, early detection and continuous monitoring of sleep disorders are beneficial. We investigated a minimalistic approach for the severity classification, severity estimation, and progression monitoring of obstructive sleep apnea (OSA) in the home environment using wearables. The recursive feature elimination technique was used to select the best feature set of 70 features from a total of 200 features extracted from polysomnograms. Then we used a multi-layer perceptron model to investigate the performance of OSA severity classification using a subset of features available from either Electroencephalography or Heart Rate Variability (HRV) and time duration of Oxygen saturation (SpO_2 level). By using only computationally inexpensive features from HRV and SpO_2, we were able to achieve an area under the curve of 0.91 and an accuracy of 83.97% for the severity classification of OSA. For estimation of the apnea-hypopnea index, an accuracy of RMSE=4.6 and R-squared value=0.71 were achieved in the test set using only ranked HRV and SpO_2 features. The Wilcoxon-signed-rank test indicated a significant change ($p<0.05$) in the selected feature values for a progression in the disease. This approach paved the way for reliable OSA monitoring using edge devices. For sleep deficiency severity estimation, we investigated the development of a mathematical model for interpretable and user-friendly objective assessment from polysomnogram data. Then we developed a regression model for the estimation of sleep deficiency severity from a fusion of wearable sensor data without requiring a polysomnogram. Monte-Carlo Feature Selection and Inter-dependency Discovery were used to select features of interest. This facilitated to reduce the impact of multi-collinearity in the features. A deep Artificial Neural Network achieved the best performance of RMSE = 5.47, with an R-squared value of 0.67 for sleep deprivation severity estimation. The developed method outperformed conventional methods; e.g., the Functional Outcome of Sleep Questionnaire and the Epworth Sleepiness Scale, for assessing the impact of sleep apnea on sleep deprivation. In addition, the method is highly suitable for integration with wearables. Finally, we developed a sensor-edge-cloud sHealth framework for implementation of the developed algorithms and conducted a year-long pilot study for evaluating the models and technology feasibility in living lab environments. Smartphones were used as the edge computing device, with pre-trained machine learning algorithms implemented in the smartphone app for computing disease-related Events of Interest (EoI) on the smartphone alone (i.e., without sharing raw data with the cloud or any other computing facility). Emphasis was given to system performance maximization using computationally inexpensive signal processing algorithms and pattern recognition techniques to avoid adversely affecting the users' smartphone performance. Spatiotemporal trends of the EoI were shared and visualized in a cloud server to facilitate personalized as well as community-wide health monitoring. Additionally, we analyzed the prospect of everyday wearables and Inkjet-Printed (IJP) Wireless Resistive Analog Passive (WRAP) sensors for physiological sensing, as they are environment-friendly and disposable. Challenges in the characterization and integration of IJP body-worn analog sensors with Internet-of-Things arise due to the effect of noises and artifacts, sensor to sensor variability, and sensor misalignment effect. A random forest based regression model was developed for estimating and longtime monitoring of core body temperature using a flexible body-worn disposable IJP WRAP temperature sensor. With 5-fold cross-validation the model achieved an RMSE=0.98, R-squared value=0.99, and mean absolute error, MAE=0.59 for core-body temperature estimation. The sHealth model thus appears to be applicable for various other physiological sensing (e.g., breathing, heart rate), which may facilitate the preliminary severity estimation, monitoring, and management of diseases and reduce subsequent associated healthcare costs.

PREFACE

During the Ph.D. study which also includes my internships, I have authored/co-authored 5 journal articles (4 first authored and 1 co-authored) and 9 conference proceedings'. Two journal articles have been published, and 3 are under review. The format of this dissertation is presented as three journal articles, where I am serving as the first author for each, as listed below.

Article 1, entitled "Development of a Minimalist Method for Early Severity Assessment and Progression Monitoring of Obstructive Sleep Apnea on the Edge," listed as Chapter 2, is in the minor revision evaluation stage in the ACM Transaction on Computing for Healthcare.

Article 2, entitled "Development A Smart Health (sHealth) Centric Method Toward Estimation of Sleep Deficiency Severity from Wearable Sensor Data Fusion," listed as Chapter 3, is under review in the BioMed Informatics Journal.

Article 3, entitled "A pilot study towards a smart-health framework to collect and analyze biomarkers with low-cost and flexible wearables at a smart and connected community," included in Chapter 4, is in the major revision evaluation stage in the Smart Health Journal.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1
INTRODUCTION

1.1 PROBLEM STATEMENT

Delayed diagnosis of disease often necessitates complicated treatment and leads to decreased quality of life for the patient, thereby increasing the healthcare burden as well as the cost of care. One way to improve healthcare delivery is by detecting and diagnosing diseases at an early stage when they are far more treatable. Although modern technology has contributed substantially to advancing healthcare in a clinical setting with the introduction of precision diagnostic tools, minimally invasive surgical procedures, remote consultation, technology-based therapeutic tools, point of care delivery, etc., early detection of disease in a pre-clinical setting at a living lab environment has yet to be achieved. Preliminary estimation of disease severity provides more valuable insights into the stage of the disease and has the potential to motivate the patient to seek medical attention in a timelier manner.

For many diseases, the introduction of pathological conditions is reflected in the vital signs; i.e., heart rate, respiration rate, oxygen saturation level, temperature, and blood pressure. In addition, bio-signals, such as the Electrocardiogram (ECG), Electroencephalography (EEG), and pulse oximetry (SpO_2), reveal a significant amount of information that helps in early diagnosis. Advances in the field of wearable body sensor networks (BSN), which is usually composed of a collection of multiple tiny wireless sensors, has made the monitoring of vital signs and capturing of biopotentials activity more reliable and user-friendly. As of now, the data collected using wearable devices are usually transmitted to a cloud computing facility for making inference with the help of artificial intelligence-enabled methods. This not only introduces a high likelihood of privacy compromise, but also increases both signal noise and the risk of false data injection in a Smart Health (sHealth) framework. Moreover, the addition of millions of Internet of Things (IoT) devices as part of smart city infrastructures will introduce a huge burden on network bandwidth and cloud computing capacity, as well as storage capacity in the near future. Edge computing offers an alternative to as well as an extension of cloud computing, where data processing and inference is conducted at the edge device near to the users. On-device data processing and machine learning help to eliminate privacy issues and reduce the IoT related bandwidth requirements. Application of edge computing for healthcare applications holds great promise, but it is still evolving. Using edge computing with widely available, user-friendly devices, such as smartwatches, smart bands, and smartphones, may help in the early detection and severity estimation of diseases; however, many challenges remain as these devices are resource-constrained due to less computing power, smaller battery capacity, and less-precise sensors.

1.2 RELATED WORKS

1.2(a) Obstructive Sleep Apnea

On average, 22% of men and 17% of women in the world-wide population experience obstructive sleep apnea (OSA) [1]. Insufficient sleep, drowsiness, and daytime sleepiness are some of the direct consequences of OSA. Untreated OSA is either responsible for or strongly correlated with many diseases, such as myocardial infarction, hypertension, high blood pressure, and depression [2]. OSA is not persistent for the entire duration of sleep; rather it appears in a repetitive manner as events that are associated with physiological changes, such as heart rate, oxygen saturation level, and respiration. The American Academy of Sleep Medicine (AASM) Taskforce defined an apnea/hypopnea event as a complete cessation or transitory reduction of breathing when the airflow decreases by more than 50% from the amplitude of baseline and the oxygen saturation level decreases by at least 4%, with the event persisting for a minimum of 10 seconds [3]. OSA diagnosis and severity assessment is currently measured using a polysomnogram (PSG), a comprehensive test that collects different types of physiological signals including, ECG, EEG, SpO_2, leg movement, body movement, and respiration. These tests are usually conducted in sleep laboratory settings and analyzed by certified sleep clinicians. Patients diagnosed with OSA are generally treated with continuous positive airway pressure (CPAP) devices. Oral appliance (OA) and surgical operations are possible alternatives to CPAP in the management of OSA [4]. Some researchers have evaluated the effectiveness of weight reduction in the early stages of OSA [5]. Although CPAP is generally equipped with a system for monitoring Apnea/Hypopnea Index (AHI), other approaches require PSG or separate arrangements for monitoring OSA progression/reduction.

Many studies have been conducted to diagnose and estimate the severity of OSA using features of the ECG and EEG [6]. For example, Liu *et al.* proposed using a neural network method for the detection of obstructive sleep apnea from the EEG [7], while Almuhammadi *et al.* proposed an efficient method for sleep apnea classification based on EEG signals [8]. Khandoker *et al.* similarly used feedforward neural networks with wavelet-based features of ECG for automated scoring of OSA [9]. De Chazal *et al.* utilized time and frequency domain measures of heart rate variability (HRV) along with ECG derived respiratory signal features to obtain a 92.5% accuracy with a Quadratic discriminant classifier [10]. Another example is that of Song *et al.*, who approached sleep apnea detection from ECG signals using hidden Markov models [11]. A final example concerns the use of a Poincaré plot of HRV to observe sleep apnea patients before and after Continuous Positive Airway Pressure (CPAP) treatment [12]. However, these investigations are limited to classification of apneic subjects from healthy subjects and do not address the important issue of continuous monitoring of an OSA patient for the evaluation of treatment effectiveness.

When OSA is severe it can lead to serious health issues, including stroke, heart failure, and even death. Hence, severe OSA requires immediate detection and medical attention. Although the use of non-polysomnographic features in OSA screening (apneic or normal) has been investigated previously, classification of severe and non-severe OSA from ECG alone is challenging. Eiseman *et al.* attempted to classify sleep apnea severity from ECG derived sleep spectrograms [13], while Park *et al.* investigated the correlation between sleep apnea severity and heart rate variability indices [14]. Investigation by Raymond *et al.* using a combination of pulse oximetry and heart rate variability via a "cardiac-oximetry disturbance index" is an example of an alternative to polysomnogram-based AHI [15]. While these studies illustrate the range of possible approaches to detecting the severity of OSA from an ECG, development of a simple yet reliable method to categorize severe OSA is sorely needed.

1.2(b) Sleep Deficiency Severity Estimation

With the increased rise of obesity, excessive usage of personal gadgets, urbanization and other socio-economic changes, sleep/wake homeostasis is becoming adversely impacted and disrupting the normal circadian rhythm. Neither body nor the brain can function properly without adequate sleep. Moreover, sleep deficiency can lead to physical and mental health problems, injuries, loss of productivity, and even a greater risk of death. Previous research suggests that complete sleep deprivation impairs attention and working memory. Moreover, it also affects related functions, such as long-term memory and decision-making. Even partial sleep deprivation can negatively impact attention and vigilance in the long run [16], as well as central auditory processing [17]. Rault *et al*. suggest further that sleep deprivation reduces respiratory motor output by altering its cortical component with subsequent reductions in inspiratory endurance [18].

Sleep deprivation is commonly estimated by using standard questionnaires along with a sleep test where a polysomnogram is captured. Questionnaire-based approaches have many limitations, including high bias, the need for a long evaluation period, etc. On the other hand, polysomnography is expensive, has limited availability, and is less user-friendly. As an alternative, a few other methods have been developed for objective assessment of sleep health. In recent years, efforts to develop and use multi-modal sensors and technologies to monitor sleep, which includes sleep patterns monitoring, wellness applications, sleep coaching of individuals with chronic conditions, etc., have greatly expanded [19]. However, most of the approaches have focused on sleep quality assessment or sleep score estimation [20-23]. Sleep deficiency provides a more comprehensive evaluation of sleep health than a single sleep score or sleep quality assessment. Sleep deficiency measures seek to capture the lack of enough sleep (sleep deprivation), not receiving all types of sleep that a human body needs, and sleep disorders that contribute to poor quality of sleep [24]. Early detection of sleep deficiency is beneficial to avoid many linked chronic health problems, including heart disease, kidney disease, high blood pressure, diabetes, stroke, obesity, and depression. A method for objective assessment of sleep deficiency from wearable sensor data only is not well-established and, thus, requires further investigation.

1.2(c) Smart Health

Our proposed Smart Health (sHealth) framework aims to deliver improved healthcare using IoT centric solutions. It is an emerging paradigm for efficient processing, sharing and visualization of healthcare data, which is emanating from different IoT devices. The Global Observatory for Electronic Health (eHealth) of the World Health Organization (WHO) has defined Mobile Health (mHealth) as a medical and public health practice supported by mobile devices, such as mobile phones, personal digital assistants, and other wireless devices [25]. mHealth has already brought about revolutionary improvements and benefits for ubiquitous and pervasive healthcare delivery. sHealth can be perceived as an upgraded and extended version of mHealth which aims to incorporate smart home and smart city infrastructure and related IoT devices including smartphones, smart bands, smartwatches and other wearables for disease diagnosis, monitoring and healthcare delivery. mHealth has mainly focused on personalized monitoring, whereas sHealth acknowledges the role of home, family, and community as important contributors to individual health and wellbeing. It thus aims to connect data, people and systems to enable long-term care rather than providing sporadic treatment to acute conditions.

S. Rani *et al.* investigated the use of smart health elements for controlling the chikungunya virus, a mosquito instinctive disease that spreads hurriedly in various parts of the country [26]. They presented an IoT-enabled model which addressed data collection from the sensors, objects, and people and gathered all of the data at the cloud to enable healthcare professionals to take preventive and controlling measures. R. K. Pathinarupothi *et al.* reported an IoT-based smart edge system for remote health monitoring, in which wearable vital sensors transmitted data into the IoT smart edge [27]. The IoT smart edge then employs a risk-stratified protocol to trigger rapid push of alerts and personal health motifs to the physicians, and also facilitate pull of detailed data-on-demand through the cloud. W. N. Ismail *et al.* proposed a convolutional neural network-based health model for regular health factors analysis in the Internet-of-Medical Things environment [28]. This proposed method uses the health conditions and lifestyle patterns related to chronic diseases collected through IoT-devices.

While cloud computing has its own benefits, such as increased computational capacity, storage facility, and high reliability making it suitable for unstructured big data landscape, it also has certain limitations due to the huge burden imposed by the data transmission on the network bandwidth. The expected addition of millions of IoT devices in the near future may well overload the computational capacity of cloud infrastructures. This is a particularly salient concern for medical applications where sharing of raw data is associated with an increased likelihood of privacy compromise, which in turn may be used by government and law enforcement bodies for surveillance purposes, thus limiting personal freedom. If widely used the chance of false data injection is also very high, which may lead to incorrect diagnosis and treatment suggestions. A. Singh, and K. Chatterjee highlighted the security issues, threats and challenges of cloud computing [29].

Fog computing offers an alternative to cloud computing where computation tasks are performed close to the user and terminal IoT devices. P. Verma and S. K. Sood investigated a Fog assisted-IoT enabled patient health monitoring scheme [30], which adopted an event triggering-based data transmission methodology to process the patient's real-time data at fog layer. This temporal mining concept has been used to analyze the events of adversity by calculating the temporal health index of the patient. J. Hu *et al.* proposed an IPv6-based framework for fog-assisted healthcare monitoring [31]. In their proposed framework the body-sensing layer generates physiological data, and the fog computing nodes in the fog layer collect and analyze time-sensitive data. While Fog-computing enables a faster response by reducing the network bandwidth requirement and computational burden of the cloud, it is subject to privacy and security breaches as the raw data is released from the individual's personal devices.

Edge-computing with on-device data processing and machine learning offers a promising solution to reduce privacy and security related issues, enable real-time disease monitoring and detection and eliminate network dependency by a sizeable amount [32-34]. However, edge computing is challenging due to the fact that edge devices are resource constrained as they are equipped with low processing capacity and have limited battery power and storage [35]. Artificial intelligence based models integrated in an edge application for automated disease detection need to be quantized, which often leads to poor performance. Minimizing the trade-off between runtime and accuracy is challenging. Hence, a hybrid edge-cloud model may offer a more advantageous solution combining both the advantages of edge computing and cloud computing [36]. Further investigation is required for edge computing based sHealth model development and deployment.

1.3 STATEMENT OF RESEARCH OBJECTIVES

1) Development of an algorithm for early severity estimation of obstructive sleep apnea on the edge and implementation of the algorithm in a smartphone application.

2) Development of an algorithm for sleep deficiency severity estimation using wearable devices in Smart Health.

3) Development of a Smart Health framework for capturing events of interest from the living lab, implementing the developed algorithms on edge devices, incorporating data processing for communitywide spatiotemporal monitoring; identifying the prospect and challenges related to on-device data processing and machine learning; and outlining prospective solutions.

1.4 TECHNICAL CHALLENGES

1) A sizeable number of technical challenges are associated with developing and implementing a smart health framework that includes but is not limited to the low-cost sensing of physiological signals, extraction of reliable biomarker, development, and implementation of artificial intelligence-based inference methods on resource-constrained devices in a way that does not hamper their performance, while permitting the sharing of information to facilitate the community while maintaining anonymity, privacy, security, etc.

2) Development of an algorithm for early severity estimation of obstructive sleep apnea obtained from non-polysomnographic measures while using wearables is challenging because of the trade-off between user comfort and quality capturing of the events of interest. If multiple sensors (i.e., ECG, EEG, acceleration, etc.) are used, user comfort is reduced; whereas use of a single sensor makes the extraction of reliable biomarkers quite difficult. Development of an optimal method that provides high accuracy while maintaining user comfort and meeting edge device constraints is challenging.

3) Development and validation of an algorithm for objective assessment of sleep deficiency using user-friendly wearable sensor data only is quite challenging. The method that is pursued needs to be suitable for longitudinal monitoring and should capture the impacts of sleep related disorders.

1.5 CONTRIBUTIONS

1) An algorithm for early severity estimation and continuous monitoring of obstructive sleep apnea using edge devices.

2) An algorithm for estimation of sleep deprivation deficiency from multi-modal wearable sensor data fusion using Monte-Carlo feature selection and inter-dependency discovery.

3) A framework of smart health which includes IJP sensor characterization and integration, a smartphone application with artificial intelligence-enabled algorithms, a web server for spatiotemporal visualization, and a data pipeline for seamless information flow in the framework.

1.6 PUBLICATIONS

Journal Articles

1. B.I. Morshed, M.J. Rahman, et al, “Inkjet-printed fully-passive body-worn wireless sensors for the smart and connected community (SCC)”, J. of Low Power Electronics and Applications, 7(4), 26, 2017.
2. M.J. Rahman, B.I. Morshed, “A Minimalist Method Toward Smart Health for Early Severity Assessment and Progression Monitoring of Obstructive Sleep Apnea”, ACM Transaction on Computing for Healthcare, Mar 2020, (*minor revision response submitted*).
3. M.J. Rahman, B.I. Morshed, “A pilot study towards a smart-health framework to collect and analyze biomarkers with low-cost and flexible wearables at a smart and connected community”, Smart Health Journal, 2021 (*major revision response submitted*).
4. M.J. Rahman, B.I. Morshed, C. Preza “A Smart Health (sHealth) Centric Method Toward Estimation of Sleep Deprivation Severity from Wearable Sensor Data Fusion” BioMedInformatics Journal (*submitted*).

Conference Proceedings

1. M.J. Rahman, B.I. Morshed “ A Novel Method for Estimation of Sleep Score Using a Deep Sequential Neural Network”, IEEE International Conference on Electro Information Technology, 2019, MI, USA.
2. M.J. Rahman, B.I. Morshed et al., “A Field Study to Capture Events of Interest from Living Labs Using Everyday Wearables for Spatiotemporal Monitoring Towards Smart Health” 42nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Jan 2020, Montreal, Canada.
3. M.J Rahman, B.I. Morshed, “SCC Health: A Framework for Online Estimation of Disease Severity for the Smart and Connected Community”, IEEE International Conference on Electro Information Technology, 2019, Brooks, SD, USA.
4. M.J Rahman, R. Mahajan, B.I. Morshed, “Exacerbation in Obstructive Sleep Apnea: Early Detection and Monitoring Using a Single Channel EEG With Quadratic Discriminant Analysis” 9th International IEEE/EMBS Conference on Neural Engineering, 2019, San Francisco, CA, USA.

5. M.J Rahman, B.I. Morshed, “Improving Accuracy of Inkjet-Printed Core Body WRAP Temperature Sensor Using Random Forest Regression Implemented with an Android App”, 2019 United States National Committee of URSI National Radio Science Meeting, Boulder, CO, USA.
6. M.J Rahman, R. Mahajan, B.I. Morshed, “Severity classification of obstructive sleep apnea using only heart rate variability measures with an ensemble classifier”, 2019 IEEE EMBS International Conference on Biomedical & Health Informatics, Las Vegas, NV, USA.

Other works published during this period (not part of this dissertation):

1. M.J. Rahman, E. Nemati et al., “Automated Assessment of Pulmonary Patients using Heart Rate Variability from Everyday Wearables”, Smart Health Journal, vol. 15, March 2020.
2. M.J. Rahman, E. Nemati et al., “Toward Early Severity Assessment of Obstructive Lung Disease Using Multi-Modal Wearable Sensor Data Fusion During Walking” 42nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), June, 2020.
3. E. Nemati, M.J. Rahman et al., “Estimation of the Lung Function Using Acoustic Features of the Voluntary Cough” 42nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), June, 2020.
4. M.J. Rahman, E. Nemati et al., “Efficient Online Cough Detection with a Minimal Feature Set Using Smartphones for Automated Assessment of Pulmonary Patients”, AMBIENT 2019, 1-7, Porto, Portugal.
5. Rahman, Md Mahbubur, Ebrahim Nematihosseinabadi, Viswam Nathan, Korosh Vatanparvar, Md Juber Rahman, Soujanya Chatterjee, Nazir Saleheen, Jilong Kuang, and Jun Gao. "Methods and systems for pulmonary condition assessment." U.S. Patent Application 16/792,057, filed February 18, 2021.

Posters and Presentations

1. M. J. Rahman and B. I. Morshed, "Investigating a Minimalistic Approach for Severity Estimation and Progression Monitoring of Obstructive Sleep Apnea at Home Using Wearables”, Research Fair, Institute of Intelligent Systems, Fall, 2019.
2. M. J. Rahman and B. I. Morshed, "Battery-less Smart Sensing for IoT: Improving the accuracy of IJP WRAP Temperature Sensor with Random Forest Regression", Research Fair, Institute of Intelligent Systems, Fall, 2018
3. M. J. Rahman and B. I. Morshed, "Severe Obstructive Sleep Apnea Classification from Single Lead ECG", Research Fair, Institute of Intelligent Systems, Fall, 2017
4. M. J. Rahman and B. I. Morshed, "Smartphone App Framework Development for Severity Ranking of Diseases", Annual Poster Competition, EECE, The University of Memphis, 2017.

REFERENCES

1. Franklin *et al.* “Obstructive Sleep Apnea Is a Common Disorder in the Population—a Review on the Epidemiology of Sleep Apnea.” *J. of Thoracic Disease*, 7.8 pp.1311–1322.PMC. Web. Oct. 2018.
2. Knauert, Melissa et al. “Clinical consequences and economic costs of untreated obstructive sleep apnea syndrome.” *World journal of otorhinolaryngology - head and neck surgery,* vol. 1,1 17-27. 8 Sep. 2015, doi: 10.1016/j.wjorl.2015.08.001
3. Epstein, Lawrence J., *et al.* "Clinical guideline for the evaluation, management and long-term care of obstructive sleep apnea in adults. ", *J. of clinical sleep medicine*, 5 no.03 pp.263-276, Jun 2009.
4. Spicuzza, *et al.*, "Obstructive sleep apnoea syndrome and its management. ", *Therapeutic advances in chronic disease,* 6, no. 5, pp.273-285, 2015.
5. Tuomilehto, *et al.* "Lifestyle intervention with weight reduction: first-line treatment in mild obstructive sleep apnea." *American journal of respiratory and critical care medicine* 179, no. 4 pp.320-327, 2009.
6. F. Mendonca, et al. "A Review of Obstructive Sleep Apnea Detection Approaches," in *IEEE J. of Biomedical and Health Informatics*, 2018.

7. D. Liu, Z. Pang and S. R. Lloyd, "A Neural Network Method for Detection of Obstructive Sleep Apnea and Narcolepsy Based on Pupil Size and EEG," in *IEEE Transactions on Neural Networks*, vol. 19, no. 2, pp. 308-318, Feb. 2008.
8. W. S. Almuhammadi, *et al.* "Efficient obstructive sleep apnea classification based on EEG signals," *2015 Long Island Systems, Applications and Technology*, Farmingdale, NY, pp. 1-6, 2015.
9. A. H. Khandoker,J. Gubbi and M. Palaniswami, "Automated scoring of obstructive sleep apnea and hypopnea events using short-term electrocardiogram recordings,", *IEEE Transactions on Information Technology in Biomedicine*, vol. 13,(6),pp. 1057-1067, 2009.
10. P. De Chazal, *et al.*, "Automated processing of the single-lead electrocardiogram for the detection of obstructive sleep apnoea,", *IEEE Transactions on Biomedical Engineering*, vol. 50, (6), pp. 686-696, 2003.
11. C. Song, *et al.*, "An obstructive sleep apnea detection approach using a discriminative hidden Markov model from ECG signals," *IEEE Transactions on Biomedical Engineering*, vol. 63, (7), pp.1532-1542, 2016.
12. C. Mermigkis, *et al.*, "Poincaréplot in obstructive sleep apnoea patients before and after CPAP treatment," *European Respiratory Journal*, vol. 34, (5), pp. 1197-1198, 2009.
13. N. A. Eiseman, *et al.*, "Classification algorithms for predicting sleepiness and sleep apnea severity,", *J. Sleep Res.*, vol. 21, (1), pp. 101-112, 2012.
14. D. Park, *et al.*, "Correlation between the severity of obstructive sleep apnea and heart rate variability indices,", *J. Korean Med. Sci.*, vol. 23, (2),pp. 226-231, 2008.
15. B. Raymond, R. M. Cayton and M. J. Chappell, "Combined index of heart rate variability and oximetry in screening for the sleep apnoea/hypopnoea syndrome," *J. Sleep Res.*, vol. 12, (1),pp. 53-61, 2003.
16. Alhola, Paula, and Päivi Polo-Kantola. "Sleep deprivation: Impact on cognitive performance." *Neuropsychiatric disease and treatment*, 2007.
17. Liberalesso, Paulo Breno Noronha, Karlin Fabianne Klagenberg D'Andrea, Mara L. Cordeiro, Bianca Simone Zeigelboim, Jair Mendes Marques, and Ari Leon Jurkiewicz. "Effects of sleep deprivation on central auditory processing." BMC neuroscience 13, no. 1 (2012): 1-7.

18. Rault, Christophe, Aude Sangaré, Véronique Diaz, Stéphanie Ragot, Jean-Pierre Frat, Mathieu Raux, Thomas Similowski, René Robert, Arnaud W. Thille, and Xavier Drouot. "Impact of Sleep Deprivation on Respiratory Motor Output and Endurance. A Physiological Study." *American journal of respiratory and critical care medicine* 201, no. 8 (2020): 976-983.
19. Perez-Pozuelo, Ignacio, Bing Zhai, Joao Palotti, Raghvendra Mall, Michaël Aupetit, Juan M. Garcia-Gomez, Shahrad Taheri, Yu Guan, and Luis Fernandez-Luque. "The future of sleep health: a data-driven revolution in sleep science and medicine." *NPJ digital medicine* 3, no. 1 (2020): 1-15.
20. E. Dafna, A. Tarasiuk and Y. Zigel, "Sleep-quality assessment from full night audio recordings of sleep apnea patients," 2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2012, pp. 3660-3663, doi: 10.1109/EMBC.2012.6346760.
21. S. Cheng and H. Mei, "A Personalized Sleep Quality Assessment Mechanism Based on Sleep Pattern Analysis," 2012 Third International Conference on Innovations in Bio-Inspired Computing and Applications, 2012, pp. 133-138, doi: 10.1109/IBICA.2012.55.
22. Sadek, Ibrahim, Antoine Demarasse, and Mounir Mokhtari. "Internet of things for sleep tracking: wearables vs. nonwearables." *Health and technology* 10, no. 1 (2020): 333-340.
23. H. M. Sajjad Hossain, Sreenivasan R. Ramamurthy, Md Abdullah Al Hafiz Khan, and Nirmalya Roy. 2018. An Active Sleep Monitoring Framework Using Wearables. ACM Trans. Interact. Intell. Syst. 8, 3, Article 22 (August 2018), 30 pages. DOI:https://doi.org/10.1145/3185516
24. Sleep Deprivation and Deficiency, National Heart Lung and Blood Institute, https://www.nhlbi.nih.gov/health-topics/sleep-deprivation-and-deficiency, Retrieved on June 25, 2021.
25. World Health Organization. “mHealth: new horizons for health through mobile technologies.”, Geneva, Switzerland: World Health Organization; 2011.
26. S. Rani, S. H. Ahmed and S. C. Shah, "Smart Health: A Novel Paradigm to Control the Chickungunya Virus," in *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1306-1311, April 2019, doi: 10.1109/JIOT.2018.2802898.

27. R. K. Pathinarupothi, P. Durga and E. S. Rangan, "IoT-Based Smart Edge for Global Health: Remote Monitoring with Severity Detection and Alerts Transmission," in *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2449-2462, April 2019, doi: 10.1109/JIOT.2018.2870068.
28. W. N. Ismail, M. M. Hassan, H. A. Alsalamah and G. Fortino, "CNN-Based Health Model for Regular Health Factors Analysis in Internet-of-Medical Things Environment," in *IEEE Access*, vol. 8, pp. 52541-52549, 2020, doi: 10.1109/ACCESS.2020.2980938.
29. Ashish Singh and Kakali Chatterjee, “Cloud security issues and challenges.”, *J. Netw. Comput. Appl.* 79, C, 88–115. 2017, doi:https://doi.org/10.1016/j.jnca.2016.11.027
30. P. Verma and S. K. Sood, "Fog Assisted-IoT Enabled Patient Health Monitoring in Smart Homes," in *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1789-1796, June 2018, doi: 10.1109/JIOT.2018.2803201.
31. Hu, J., Wu, K., & Liang, W., “An IPv6-based framework for fog-assisted healthcare monitoring.”, *Advances in Mechanical Engineering*. https://doi.org/10.1177/1687814018819515.
32. “Mobile-Edge Computing Introductory Technical White Paper,” ETSI. [Online]. Available: https://portal.etsi.org/Portals/0/TBpages/MEC/ Docs/Mobile-edge Computing - Introductory Technical White Paper V1%2018-09-14.pdf
33. Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo and J. Zhang, "Edge Intelligence: Paving the Last Mile of Artificial Intelligence with Edge Computing," in *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738-1762, Aug. 2019.
34. J. Chen and X. Ran, "Deep Learning with Edge Computing: A Review," in *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655-1674, Aug. 2019.
35. W. Shi, J. Cao *et al.*, “Edge Computing: Vision and Challenges,” *IEEE Internet Things Journal*, vol. 3, no. 5, pp. 637–646, Oct. 2016.
36. B. A. Mudassar, J. H. Ko, and S. Mukhopadhyay, “Edge-cloud collaborative processing for intelligent internet of things,” in Proc. the *55th Annual Design Automation Conference* (DAC 2018), pp. 1–6, 2018.

Chapter 2

DEVELOPING A MINIMALIST METHOD FOR SEVERITY ASSESSMENT AND PROGRESSION MONITORING OF OBSTRUCTIVE SLEEP APNEA ON THE EDGE

2.1 INTRODUCTION

SMART health (sHealth) aims to deliver better healthcare for the citizens of smart cities by providing Artificial Intelligence (AI) enabled Internet-of-Things (IoT) centric solutions. Cloud computing with enhanced capabilities for processing power and storage offers great advantages and benefits regarding in-depth data analysis in terms of speed, cost, data backup, and flexibility. However, it also has a number of limitations, including network outage, latency, security, privacy, and vulnerability to attacks. Specifically, sharing sensitive raw medical data on the cloud is always risky and requires obfuscation in many cases to protect a user's privacy that may inherently lead to the deterioration of data quality [1-2]. In previous studies end-users have expressed concerns regarding data sharing with the cloud [3]. Moreover, the addition of a huge number of IoT devices in the coming years is posing a serious challenge to the network capacity and processing power of the cloud infrastructures. Edge computing is an attractive alternative to cloud computing, one that aims to host computation tasks near to the data sources and end-users [4-5]. Examples of edge computing frameworks currently being developed include Intel Open VINO, Microsoft Azure IOT Edge, NVDIA EGX, etc. [6]. Many researchers are investigating the prospect of using the IoT Edge-Cloud where data processing and inference are conducted in the edge and the result is shared with the cloud [7]. Because edge computing holds tremendous promise for smart healthcare applications, development of methods suitable for edge devices is a priority [8].

Obstructive Sleep Apnea (OSA) is a very common sleep disorder among the elderly population throughout the world [9-10]. OSA arises due to a blockage or partial obstruction in the upper airway which causes repeated episodes of paused or shallow breathing during sleep, which is termed apneas. OSA is chronic and if not treated properly, may lead to serious health issues including hypertension, coronary artery disease, excessive daytime sleepiness, headache, stroke, etc. [11]. It is estimated that 26 % of US adults between the ages of 30 and 70 years have sleep apnea. Unfortunately, 80 % of the patients remain undiagnosed [12]. Usually, sleep apnea symptoms are first observed and reported by a partner of the patient when the symptom is obvious; i.e., gasping or choking and the patient is already in a moderate to severe stage of sleep apnea. The standard medical practice for the diagnosis of OSA is based on a polysomnogram captured during a sleep test. Polysomnography is expensive, complex, and involves simultaneous monitoring of multiple physiological parameters, such as heart rate, body position, respiratory variables, brain wave activity, eye movements, leg movements, and oxygen saturation. To accommodate a sleep-test a standard facility with sophisticated equipment, sleep technologists, and physicians trained in sleep medicine is recommended; hence its availability is limited.

Moreover, polysomnography is not suitable for daily use as it is performed in clinical settings. At an early stage, many patients with sleep apnea may not be aware of their condition and consequently are not willing to undergo a costly sleep test. These and related issues result in a large portion of patients with OSA going undiagnosed. Young et al. long ago reported the prevalence of undiagnosed sleep-disordered breathing to be fairly high among adult males and higher than expected among females, constituting a sizeable public health burden [13]. Edge deployment has high potential to aid in the passive pre-screening of sleep apnea. Additionally, as edge computing enables on-device processing and machine learning which reduces privacy concerns, more patients are likely to use this pre-screening method thereby increasing the likelihood of detecting sleep apnea which heretofore may have gone undiagnosed.

Classification of OSA from healthy subjects using non-polysomnographic measures has been investigated by many researchers, with a nocturnal Electrocardiogram (ECG) signal likely being the most well investigated for detecting apnea [14-19]. Electroencephalography (EEG) and respiration signals are also commonly used for detecting OSA [20-21]. Studies utilizing these approaches, which are summarized in Table 2.1 below, however, have provided limited usability information as they typically have not addressed estimation of OSA severity, a matter of great interest to users. The gold standard for OSA severity staging is based on the Apnea-hypopnea Index (AHI), which is defined as the number of apnea-hypopnea events per hour. OSA severity is defined as Normal for AHI values < 5; Mild for AHI ≥ 5 and < 15; Moderate for AHI ≥ 15 and ≤ 30, and Severe for AHI > 30/hour [22]. While the estimation of AHI is preferred when using polysomnographic measures, development and use of alternative measures to polysomnogram-based AHI estimation for different applications is a growing trend [23-27]. For example, Papini et al. investigated the use of optimized cardio-vascular features for the estimation of AHI in a heterogeneous population [28], while Grenèche et al. examined the relationship between OSA severity and the subsequent waking EEG spectral power [29].

Although some of the previous studies hold prospects for in-home severity assessment of OSA, they did not report comparative data obtained when using standard sensor-based features. This state of affairs creates a dilemma for consumers in deciding which is the optimal simplistic method for monitoring OSA. Moreover, the current ECG-based methods are mainly based on ECG derived from respiration that requires capturing ECG signal but is not possible during sleep [25][28]. Smartwatches and smart bands available in the market can report instantaneous heart rate continuously, but active data collection is required for ECG [30]. Another limitation of previous studies is the absence of verifying the integrity of features used to monitor the progression of OSA, as well as what trends to expect before an exacerbation. Moreover, most of the previous research focused on i) offline analysis, where data are collected using wearables and then exported to a PC for analysis and/or ii) cloud-based solutions where collected data is exported to the cloud for analysis [31-33]. While these types of approaches maximize accuracy, they require use of complex algorithms or features that are computationally expensive. Our focus is on the development of a minimalistic method that is simple and computationally efficient, but one that is also reasonably accurate. This approach has increased potential for being integrated with an edge application for subsequent deployment on smartphones/wearables, wherein data can be collected using various everyday consumer wearables for making more informed inferences. Edge computing provides more privacy-preserving pre-screening of OSA, can be deployed at a wider scale, and has the added benefit of being popular with consumers [34]. For these reasons and more, we investigated the severity classification and AHI estimation of OSA using only

TABLE 2.1: Comparison of minimalist method with previous studies for OSA severity

Ref.	Dataset	Sensor/ Features	Severity Classification	AHI Estimation	Comparative evaluation	Progression monitoring
M. Wu et al. [24]	150 subjects	Age, BMI, Blood pressure, Epworth sleepiness scale, etc.	Accuracy=75.6%	RMSE=16.61	NA	NA
Jung et al. [25]	226 records	ECG derived respiration cycles during sleep onset	Correct classification ratio=85.7%	MAE=3.56	NA	NA
Saha et al. [26]	50 subjects	Accelerometer, Breathing sounds, pulse oximeter	Sensitivity=92.3% Specificity=91.8%	R2 = 0.93	NA	NA
J. Jin et al. [27]	5 subjects	MEMS sensor/nasal airflow	Sensitivity=100% Specificity=85.9%	OSR=10-20%	NA	NA
Papini et al. [28]	262 subjects	ECG/ respiratory event epochs	Accuracy=82%	MAE=14.74	NA	NA
J. Grenèche et al. [29]	12 subjects	Waking EEG spectral power	-	r = 0.66	NA	NA
This work	500 subjects	HRV and SpO_2	AUC=0.91 F-1 score= 93.24%	RMSE=4.6, R2=0.74	Yes	Yes

heart rate variability and time duration measures of SpO_2 level, which are more easily and reliably captured. Table 2.1 summarizes select studies discussed above.

The accomplishments for this study are summarized as follows:

i) Identified and ranked the physiological measures and sleep characteristics that showed significant differences between OSA severity categories.

ii) Compared classification performance using features from different sensors to select the sensors suited for a minimalistic method.

iii) Investigated the severity estimation of OSA using only HRV features that can be extracted from a continuous heart rate signal, independent of other features that require ECG recordings.

The results provide support that a comparable performance can be achieved with a minimalistic method for early severity estimation and progression monitoring of OSA using only wristband data and a smartphone app.

2.2 MATERIALS AND METHODS

2.2.1 Data Set and Study Description

The Sleep Health Heart Study (SHHS) was implemented as a multi-center cohort study in two phases by the US National Heart Lung & Blood Institute and the resultant dataset is available from the National Sleep Research Resource [35]. The data were collected over 5 years, in two phases, from 9 different research centers across the entire United States and includes males and females, and patients from all major races; i.e., African-American, Asian, Caucasian, Hispanic, and Native Americans. This dataset is well planned and incorporates a number of variabilities that may be encountered in a real-world setting. Unattended home polysomnograms were obtained for both phases of SHHS by certified and trained technicians. The polysomnogram data was saved in the European Data Format (EDF), with the data processing and initial scoring completed by Compumedics software (Compumedics Ltd., Australia). Detailed manual scorings were included to annotate the database with AHI, respiratory disturbance index, sleep stages, event start and end time identification, etc. We opted to use a dataset of 500 subjects (1000 records) containing high quality data for both phases of SHHS that is available from the dataset provider as a sub-set and is recommended for use in research studies [36]. We included subjects who were healthy or diagnosed as OSA, but excluded those with central apnea for developing our classification and regression models. The distribution of subject records in the dataset is as follows: normal- 507, mild and borderline-303, and moderate to severe -190. For OSA progression monitoring, we identified 298 subjects from SHHS who showed an exacerbation; i.e., progression to higher severity stages or an increase in AHI by 10 or more during the study period. These 298 subjects constituted the exacerbating group, while another 200 subjects not showing a progression in OSA (i.e., an increase in AHI < 3) served as our control or stable group.

2.2.2 Autonomic Nervous System and Obstructive Sleep Apnea

The autonomic nervous system (ANS) is a part of the peripheral nervous system and plays an important role in balancing the internal environment of the body, which includes heart rate, blood pressure, body temperature, coughing and sneezing, sexual arousal, oxygen and Co2 level in the blood, etc. The regulations enforced by ANS take place involuntarily and without conscious effort. The autonomic nervous system has two main divisions- sympathetic and parasympathetic. Many organs are primarily controlled by either the sympathetic or parasympathetic division. The sympathetic division prepares the body for dealing with stressful or emergency situations, whereas the parasympathetic division prepares the body for rest and digestion. Changes in ANS activity are observed during normal sleep [37]. During REM sleep, a higher level of sympathetic activity is observed. On the other hand, parasympathetic activity increases during non-REM sleep, which in turn causes the heart rate and systolic blood pressure to decrease. This is especially evident in stage IV of non-REM sleep [38]. However, arousals from non-REM sleep are associated with transient rises in heart rate and blood pressure ,which are markers of the rise in sympathetic activity and withdrawal of parasympathetic tone. Arousals also induce the production of “K” complexes in the EEG. A person with sleep apnea does not receive enough air into the lungs during an apnea event. The American Academy of Sleep Medicine (AASM) defines an apnea/hypopnea event as a complete cessation or transitory reduction of breathing when the airflow decreases by more than fifty percent ($>50\%$) from the amplitude of baseline and the oxygen saturation level decreases by at least four percent ($\geq 4\%$), with the event persisting for a minimum of ten seconds (≥ 10 sec) [23]. This causes an acute deficiency in the blood oxygen level, along with a concomitant increase in carbon dioxide. To prevent death, the brain “wakes up” and resumes breathing. Because an apnea event usually happens repetitively, a patient with OSA is exposed to recurrent episodes of hypoxemia, arousals from sleep, and an increased hemodynamic stress level. The initial part of the apneic period is characterized by bradycardia (heart rate slows down) and a rise in blood pressure from baseline. The point of arousal, on the other hand, is associated with a transient surge in heart rate and blood pressure, which are markers of increased sympathetic activity and reduced parasympathetic tone. The termination of apnea is associated with a fall in cardiac output [39]. Overall, a person experiencing sleep apnea has a higher level of sympathetic activity during sleep than during wakefulness, which is opposite to a healthy person who has a lower level of sympathetic activity during sleep.

2.2.3 Pre-Processing and Feature Extraction

The recording montage for a polysomnogram consists of data from 14 channels, which include ECG, EEG, electrooculogram (EOG), electromyogram (EMG), nasal airflow, thoracic and abdominal movement signal, SpO_2, sleep hypnogram, etc. Hardware filters are used for preliminary noise reduction. The cutoff frequency for hardware filters are as follows: ECG-0.15 Hz, EOG-0.15 Hz, EMG-0.15 Hz, EEG-0.15 Hz, thoracic respiration signal-0.05 Hz, and abdominal respiration signal-0.05 Hz. The sampling rate is 125 Hz for EEG, ECG, and EMG signals, and 50 Hz for EOG. In investigating the minimalistic approach, we considered the use of features from the ECG, EEG, SpO_2 signals as the sensors for these signals are more user-friendly and widely used. The ECG signal is processed for R-peak detection by the Pan-Tompkins algorithm [40], which uses signal preprocessing with a low pass and high pass filter to remove baseline wander and muscle artifact, a moving average for minimizing noise, and an adaptive threshold for detecting the R-peaks. For R-R interval correction we used maliks rule followed by a cubic interpolation for the determination of Normal-to-Normal (NN) intervals [41]. The signal processing steps for ECG to derive the NN interval is shown in Figure 2.1 below.

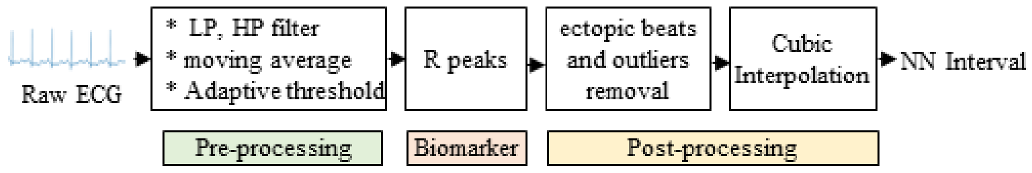


Figure 2.1: Signal processing for HRV feature extraction.

From the NN interval series we extracted time domain and frequency domain features using the HRV Toolkit available from Physionet [42]. The time-domain and frequency-domain features of the HRV are defined in Tables 2.2 and 2.3 respectively.

TABLE 2.2: Time domain measures of HRV.

Feature	Description
AVNN	Mean of the NN-interval.
SDNN	Standard deviation of all NN intervals.
SDANN	Standard deviation of the averages of NN intervals in all 5 min segments of the entire recording.
RMSSD	Defined as the square root of the mean of the squares of differences between adjacent NN intervals.
SDNN index	Mean of the standard deviation of all NN intervals for all 5 min segments of the entire recording.
SDSD	Defined as the standard deviation of the differences between adjacent R-R intervals.
pNNx	NNx count divided by the total no. of all NN intervals where NNx is the number of pairs of adjacent NN intervals differing by more than x ms.

TABLE 2.3: Frequency domain measures of HRV.

Feature	Units	Description	Frequency Range (Hz)
VLF	s2	Power in very low frequency range	≤ 0.04
LF	s2	Power in low frequency range	0.04 – 0.15
HF	s2	Power in low frequency range	0.15 – 0.4
Total Power	s2	Variance of all NN intervals	≤ 0.4
LFnu	n.u.	LF power in normalized units	-
HFnu	n.u.	HF power in normalized units	-
LF/HF	-	Ratio of LF/HF	-
LFV	-	Ratio of LF to (LF+HF)	-
HFV	-	Ratio of HF to (LF+HF)	-

For the power spectrum estimation, we used Lomb's periodogram method. The entire ECG record was divided into 5-minute epochs to estimate short-term components of HRV. In addition to the estimates of long-term and short-term HRV components, we considered another variant of HRV, which we termed as the dispersion measure of short-term HRV computed using the standard deviations of short-term HRV over the entire record. Thus, for each HRV measure, 3 variants were computed and each variant was considered a different feature for the classification and regression purpose. In total 54 HRV features were extracted. From SpO_2 we considered only the time duration related features based on the level of SpO_2 as described below [43-44]:

pctlt90- percentage of sleep time with SpO_2 level below 90%
pctlt85- percentage of sleep time with SpO_2 level below 85%
pctlt80- percentage of sleep time with SpO_2 level below 80%
pctlt75- percentage of sleep time with SpO_2 level below 75%

The EEG signal was collected using two channels from the central region of the brain-- C4-A1 and C3-A2. As the power spectral densities for these two channels are very similar, we elected to use only the signal from the C4 channel. Of note, this channel was designated as the primary EEG channel in SHHS. EEG spectral analysis was performed using the SpectralTrainFig App in MATLAB [45]. We extracted 24 spectral band features from the decontaminated EEG signal, as shown in Table 2.4, which includes rapid eye movement (REM) power, non-rapid eye movement (NREM) power, and total power at each frequency band. Also, 102 EEG spectral features (REM, N-REM power at single frequencies) were computed for 51 frequencies from 0 to 25 Hz, with a 0.5 Hz gap; i.e., 0 Hz, 0.5 Hz, 1 Hz, 1.5 Hz, …., 24.5 Hz, 25 Hz.

TABLE 2.4: Spectral band measures of EEG.

Frequency Band	Hz	Features
Slow Oscillations	0.5-1	Total Power, REM and NREM Sleep Power
Delta	1-4	Total Power, REM and NREM Sleep Power
Theta	4-8	Total Power, REM and NREM Sleep Power
Alpha	8-12	Total Power, REM and NREM Sleep Power
Sigma	12-15	Total Power, REM and NREM Sleep Power
Slow Sigma	12-13.5	Total Power, REM and NREM Sleep Power
Fast Sigma	13.5-15	Total Power, REM and NREM Sleep Power
Beta	13-30	Total Power, REM and NREM Sleep Power

In total, 200 features were extracted from the polysomnogram as follows: Sleep characteristics-14, HRV-54, SpO_2-4, EEG-126, and blood pressure (BP)-2. Sleep characteristics included sleep efficiency, sleep latency, sleep time, wake after sleep onset (WASO), time in sleep stage 1 (timest1), sleep stage 2(timest2), sleep stage 3-4(timest34); the percentage of time in sleep stage 1 (timest1p), sleep stage 2 (timest2p) and sleep stage 3-4 (timest34p); and total time in REM sleep, total time in N-REM sleep, total time in bed, and sleep disturbance index. Sleep characteristics were extracted based on the polysomnograms using Compumedics Software (Compumedics Ltd, Australia), as done when the SHHS data set was prepared. The block diagram for feature extraction is shown in Figure 2.2 below.

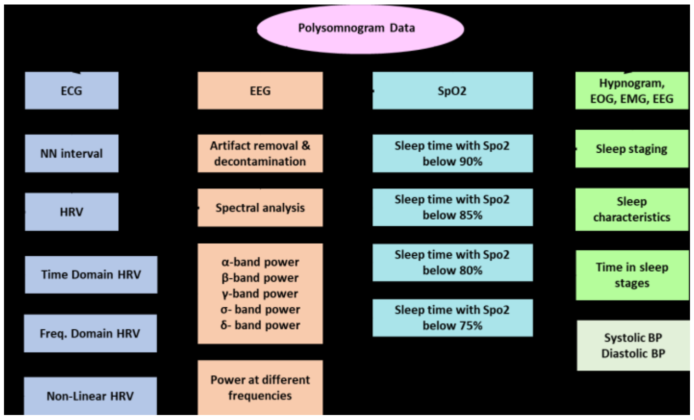


Figure 2.2: Block diagram for feature extraction.

2.2.4 Statistical Analysis, Feature Selection, and Classification

All of the OSA records from the dataset were previously divided into two groups. The first group included subjects having moderate to severe (MODS) sleep apnea; i.e., AHI>=15, while the second group consisted of those individuals with borderline (AHI ~ 5) and mild OSA (AHI <15). The MODS group had a median age of 70 and a median BMI of 29.6, whereas the mild group had a median age of 67 and a median BMI of 28.6. The variables in the dataset were examined for normality using the Shapiro-Wilk test. Because most of the variables failed to pass this normality test, the Mann-Whitney U test was used to determine if significant differences existed between the anthropometric parameters and sleep characteristics of the MODS OSA patients when compared to the other subjects. HRV and EEG features were compared between the two groups using the Mann-Whitney U test, as well. Boxplots were used to visualize the mean ranks of the HRV and EEG features of the two groups. Min-Max normalization was used for all of the features to facilitate the boxplot comparison. The entire dataset was further divided into training (comprising 80% of the data) and test (the remaining 20% of the data) sets for feature selection and classification. The data were reshuffled before making the random split into the training and test sets. For feature selection, the Recursive Feature Elimination (RFE) technique was employed [46], as it eliminates collinearity and dependencies by removing a small number of features per iteration recursively. To find the optimal number of features, repeated k-fold cross-validation was used with the number of folds=10 and the number of repeats=5. The outer resampling method was used to minimize selection bias [47]. Accuracy was used as the performance metric, with the Random Forest being used as the external estimator (wrapper) in the feature selection process. The Caret package from R was used to implement the feature selection process [48].

We investigated different classifiers viz. Logistic Regression, Random Forest, Ada-Boost, Support Vector Machine (SVM), and Multi-layer Perceptron (MLP) for the classification of MODS OSA category patients from the mild OSA. The grid search method was used for tuning the hyper parameters when applicable. Scikit-learn and Keras were implemented as the model development environment [49-50]. All feature values were standardized using centering and scaling; i.e., z-score normalization before feeding into the classifier. As described in section 2, the study dataset contained 303 records in the mild and borderline category compared to 190 records in the moderate to severe category. The latter grouping was up sampled to make the number of instances equal to the majority class as classification using class-imbalanced data is biased in favor of the majority class [51]. However, the SMOTE method was applied to the training set only, as the test set was not up sampled so that real-world performance could be replicated during the test. To make the dataset balanced, synthetic minority oversampling (SMOTE) was used [52]. Initially, classification was performed using all of the ranked features. The classification performance of the minimalistic approach was then investigated by using ranked features from only i) HRV and SpO_2 and ii) EEG. MLP superseded the performance of other classifiers. The configuration of the MLP was as follows: number of hidden layers=1, number of units in hidden layers=30, number of epochs=100, learning rate=0.1, and batch size=20. Binary cross entropy was used as the loss function and relu was applied as the activation function for the hidden layer, with sigmoid used in the output layer. A Stochastic Gradient Descent (SGD) algorithm was used as the optimizer and to avoid overfitting L2 weight regularization was used with alpha=0.0001 [53]. In addition, the Early stopping criterion was used to avoid overtraining. A high-level block diagram for feature extraction, feature selection, and classification is shown in Figure 2.3 below.

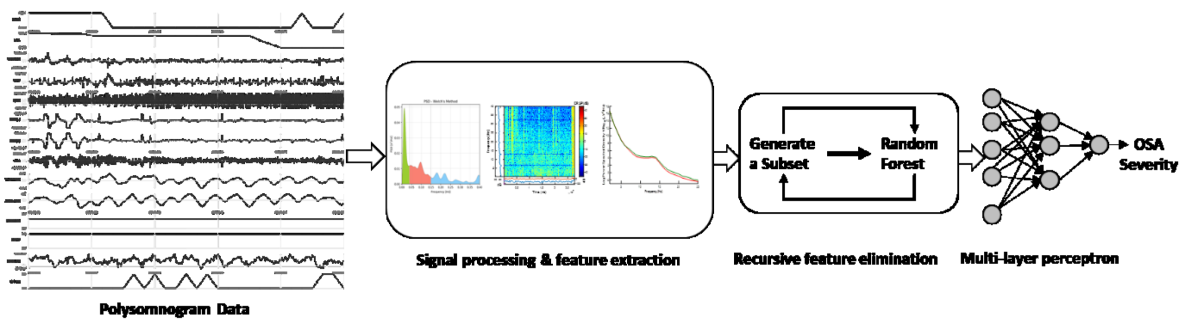


Figure 2.3: High-level diagram for OSA severity classification from polysomnogram data.

2.2.5 Regression for Severity Estimation

In order to examine the validity of estimating OSA severity from non-polysomnographic features using a minimalistic approach, we calculated the value of AHI using the ranked features from HRV and SpO_2 only. In the pre-processing stage, Cook's distance, defined in equation 2.1 below, was used for identifying HRV outliers in the dataset [54].

$$D_j = \frac{\sum_{k=1}^{n}\left(\widehat{y_k} - \hat{y}_{k(j)}\right)^2}{pMSE}$$

Equation 2.1.

In this equation, y_k is the kth fitted response value, $y_{k(j)}$ is the kth fitted response value excluding observation j from the fit, MSE is the mean squared error, and p is the number of coefficients in the regression model. Observations with Cook's distance larger than three times the mean Cook's distance were likely an outlier and, hence, were discarded. The remaining dataset was reshuffled and divided into training (70%), validation (15%), and test (15%) sets for validating and testing the regression models. For developing the regression model, MLP neural networks with backpropagation were investigated. The finalized MLP model had a single hidden layer with 30 units and used the scaled conjugate gradient algorithm for training [55]. For the performance evaluation, root mean squared error (RMSE) and the coefficient of determination (R^2) were used in addition to residual analysis. Feature values were standardized before fitting into the regression models.

2.2.6 Progression Monitoring of OSA

The distribution of patients in different severity categories before and after exacerbation were visualized using pie charts. The Wilcoxon-signed rank test was applied to determine if the values of the ranked features changed significantly before and after the exacerbation. The same features were tested for the control group and exacerbation group and the features showing significant changes only for the exacerbation group were identified. Changes for the exacerbating group were visualized using interval plots.

2.3 RESULTS

The results of the Mann-Whitney U test for the physiological and sleep parameters of the mild and MODS OSA groups are shown in Table 2.5, where measures showing a significant difference ($p \leq 0.05$) between the two groups are marked with an asterisk. The MODS group had a significantly higher OAHI, median Systolic blood pressure, Neck20, total time in sleep, number of arousals, and wake times after sleep onset than the mild group. The findings for sleep efficiency and % time in sleep revealed a different pattern, with the Mild group showing significantly higher values when compared to the MODS group.

TABLE 2.5: Results of the Mann Whitney U test between the median values for the mild and moderate-to-severe OSA groups.

Parameter	Mild group N = 303	MODS group N = 190	p-value
OAHI	7.95	25.89	0.001*
Syst BP	123.00	128.00	0.017*
Dias BP	72.00	72.00	0.702
Neck20	39.00	39.90	0.037*
Sleep time	364.25	364.00	0.755
Sleep latency	16.50	16.00	0.716
Sleep efficiency	83.56	80.00	0.001*
% time in sleep	16.81	12.46	0.002*
Time in sleep	57.86	62.00	0.002*
Arousal	17.36	25.18	0.001*
Wake after sleep	52.00	73.00	0.001*

The Mann Whitney U test for the HRV and SpO_2 features also revealed significant differences between the two groups. The bar diagram comparison for the HRV and SpO_2 feature ranks between the groups based on the Mann U test is shown in Figure 2.4. Only the features showing significant differences ($p \leq 0.05$) are included in the plot.

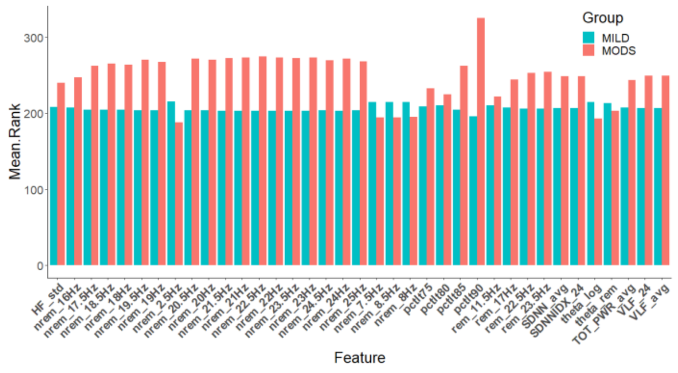


Figure 2.4: Bar diagram comparison between OSA severity categories based on Mann-Whitney U test for HRV, SpO_2, and EEG features.

Feature ranking using the recursive feature elimination technique provides the optimal number of features as 70 to achieve the best accuracy. Figure 2.5(a) shows the results of the recursive feature elimination method. Adding more features did not improve the cross-validation accuracy. The top 20 features with their relative importance are listed in Figure 2.5 (b). The top features are from different signal sources and include EEG, HRV, SpO_2, and sleep hypnogram (for sleep stages).

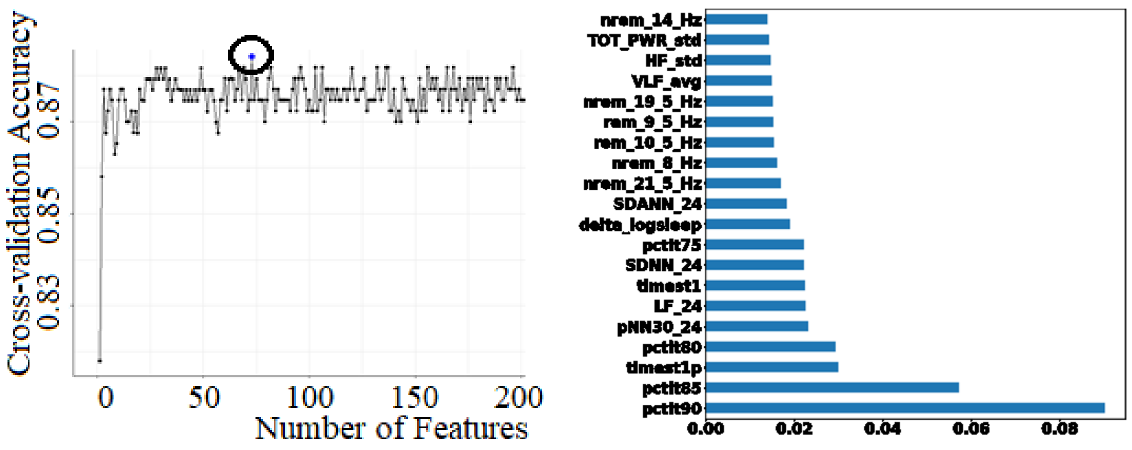


Figure 2.5: (a) Results of feature ranking and (b) Top-ranked 20 features.

The performance of different classifiers using the 70 features selected by the RFE method are shown in Table 2.6. The MLP model performed the best with an accuracy of 93.24%, precision of 93.24%, recall of 93.24%, and F-1 score of 93.24%. Based on the performance, MLP was selected as the classification model for investigating the performance of the minimalistic approach.

TABLE 2.6: Five-fold cross-validation results for different classifiers

Classifier	Apnea Severity Classification		
	Precision	Recall	F-1 score
Logistic Regression	83.9%	83.79%	83.77%
Random Forest	91.31%	90.54%	90.34%
AdaBoost	93.39%	93.41%	92.04%
SVM	86.27%	85.98%	85.95%
Multi-layer Perceptron	93.24%	93.24%	93.24%

Fig. 2.6 shows the learning curve and the area under the receiver operating characteristics (roc) curve. With increase of training data size performances in both training and test sets are increased. Table 2.7 shows the classification performance of the minimalistic approach with that of the ranked polysomnographic measures. There is a reduction in The classification performance accuracy of the minimalistic approach was slightly reduced when compared with the complete data set. Nonetheless, these findings indicate that a reasonable degree of accuracy can be achieved with the minimalistic approach at the gain of comfort, reduced cost, and user-friendliness.

Table 2.7: Comparison of minimalist method with polysomnogram

Classifier	Apnea Severity Classification	
	Training Set	Test Set
All Ranked Features (70)	96.45%	93.24%
Ranked HRV and SpO_2 Features (24)	88.92%	83.97%
Ranked EEG features (42)	82.60%	77.02%

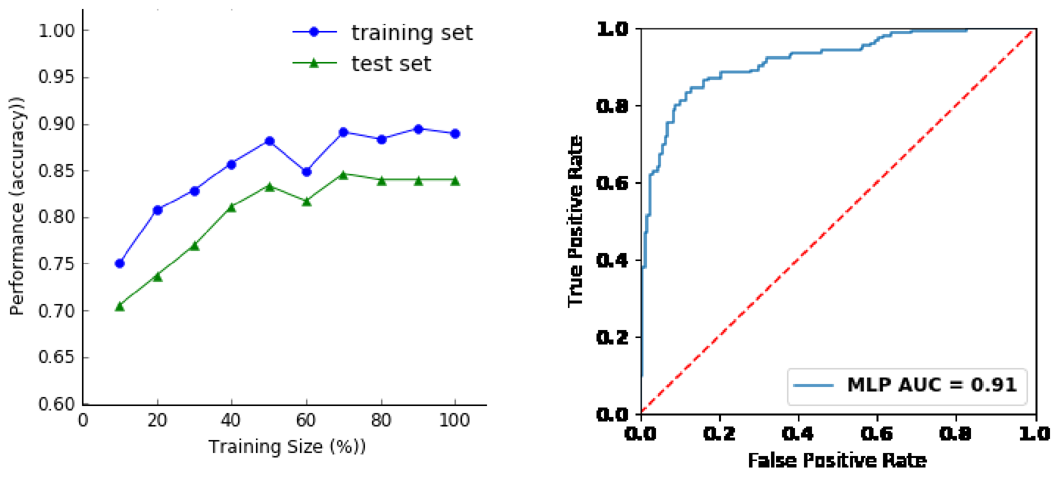


Figure 2.6: a) Learning curve and b) area under the roc curve.

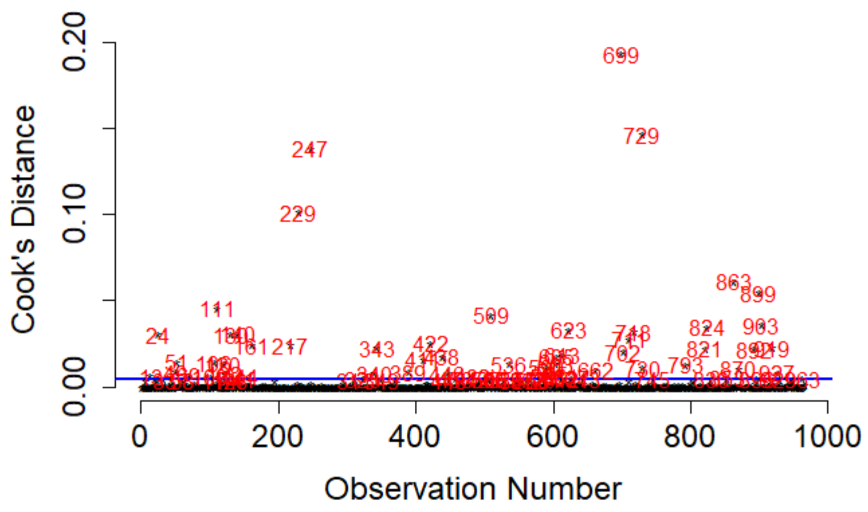


Figure 2.7: Case order plot for Cook's distance.

Identification of influential observations in the data set through the Cook's distance method indicates the presence of outliers in the data, as displayed in Figure 2.7, which shows the case order plot of Cook's distance. The observations outside the recommended threshold (indicated by the blue line in the figure) are influential. The performance of the MLP regression model using the ranked HRV and SpO_2 features for the estimation of OSA severity (i.e., AHI) is shown in Figure 2.8. The line-of-fit shows a good fit with most of the points around the ideal line. The achieved r-squared values in the training, validation, and test sets were 0.83, 0.74, and 0.71, respectively. The root mean squared error of the regression model in the test set is 4.6. As shown in Figure 2.9, the best validation performance was achieved at epoch 7. The error histogram for the regression model, displayed in Figure 2.10, reveals that the majority (~300) of the observations fell within the error bin of -2.07 and the zero-error bin.

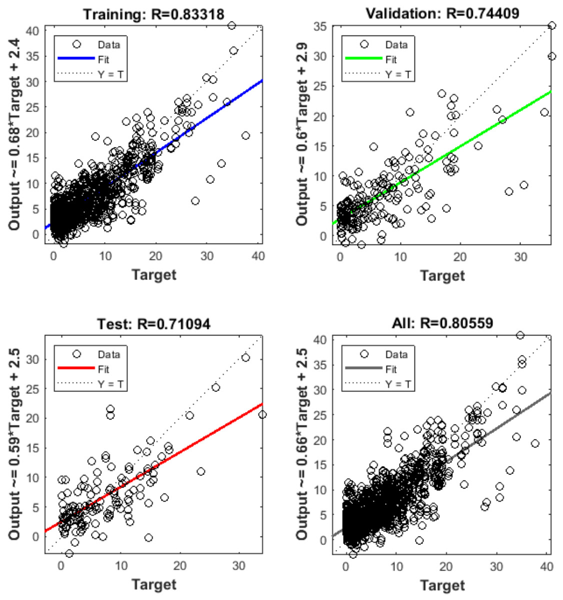
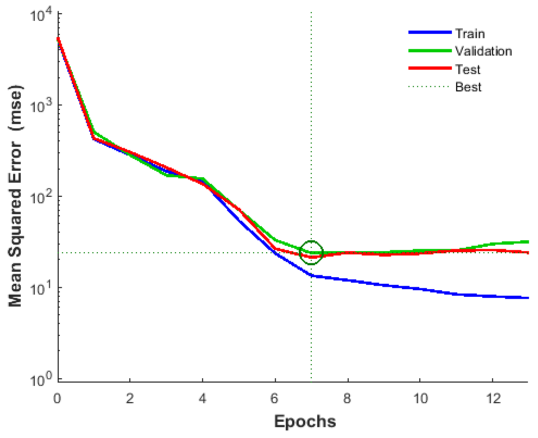


Fig. 2.8: Plot of fit for the regression model



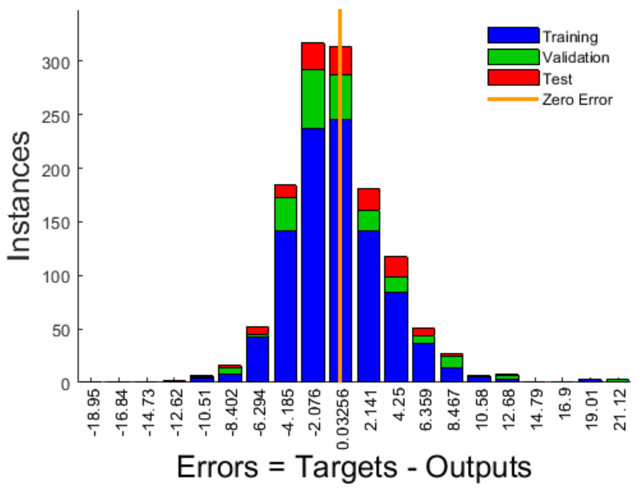


Figure 2.9: Performance curve for training, validation, and test sets.

Figure 2.10: Error histogram for the regression model.

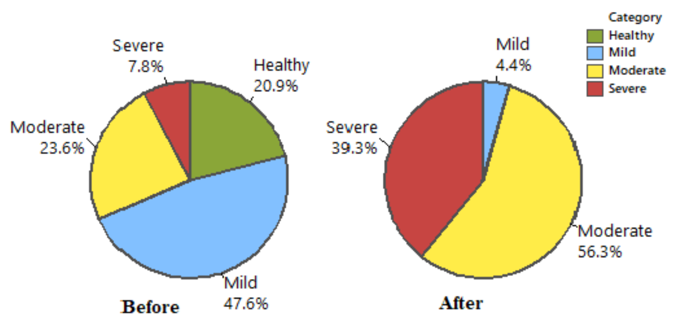


Fig. 2.11: Subject distribution of the exacerbating group during baseline and follow-up visit

Figure 2.11 shows the pie chart for the distribution of subjects in different severity categories before and after an exacerbation. The proportion of the subjects in the severe category increased to 39.3% from 7.8%, in the moderate category from 23.6% to 56.3%, whereas the proportion of subjects in the mild and borderline category was markedly reduced.

TABLE 2.8: Comparison of time and frequency domain HRV measures at baseline and exacerbation stage

	Baseline Median (IQR)	Exacerbation Median (IQR)	Wilcoxon SRT	Normality
Mean RR	917.00(171.92)	900.52(150.25)	0.043	P<.001
pcNN30	30.89(32.36)	28.26(30.78)	0.002	P<.001
pcNN50	14.72(25.00)	12.34(20.88)	0.001	P<.001
RMSSD	40.06(33.45)	37.45(29.75)	0.006	P<.001
SDANN	63.56(31.68)	67.37(30.93)	0.049	P<.001
ULF	1683(1991)	1937(2054)	0.023	P<.001
VLF	1467(1669)	1680(1766)	0.009	P<.001
HF	276.1(557.0)	225.0(448.0)	0.004	P<.001
HFV	0.063(0.913)	0.048(0.06)	.001	P<.001

A co-variate analysis controlling for age and BMI showed significant ($p<0.05$) partial correlations of HRV and SpO_2 features with AHI. The average increase in AHI due to the progression in disease was 20 AHI. The Wilcoxon-signed-rank test was used to determine if the changes observed for both the exacerbating and control group (stable subjects) were significant. While the HRV features showed significant changes ($p<0.05$) for both groups, only those for the exacerbating group are shown in Table 2.8. The change in individual feature mean values along with 95% confidence intervals for the selected physiological, HRV, SpO_2, and EEG features are visualized in Figure 12(a), 12(b), and 12(c).

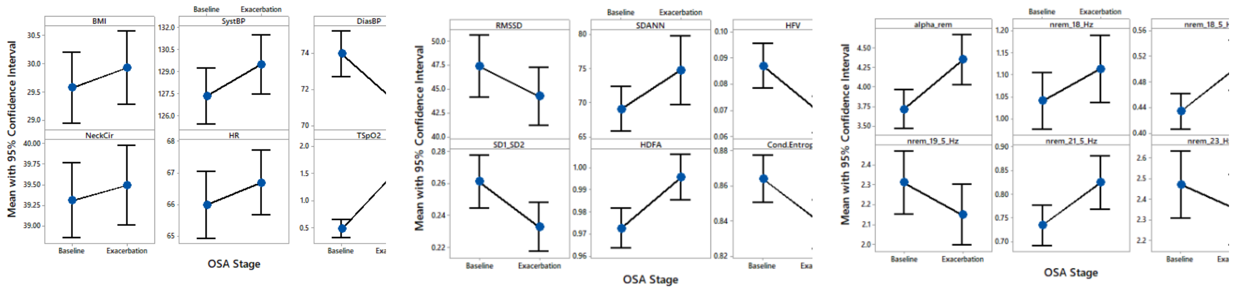


Figure 2.12: Interval plots for a) for physiological measures and b) HRV features c) EEG features at Baseline and Exacerbation.

2.4 DISCUSSION, LIMITATIONS, AND FUTURE WORK

Preliminary estimation of apnea severity and progression monitoring at home using the minimalistic approach was not intended to serve as a replacement for the polysomnogram. Rather it is intended for identifying individuals who have not been diagnosed for OSA and have never undertaken a polysomnogram in the sleep laboratory or even pursued a home sleep test. We mention this as we are well aware of the potential risk of underestimation of OSA severity when home assessments are used [56]. In future work one goal will be to estimate the HRV features from the PPG signal captured by a smartwatch or smart band as shown in Figure 2.13 [57]. This type of minimalistic approach has the potential to offer an edge computing-based solution to OSA detection and monitoring and may be more suitable for implementation on everyday consumer-grade wearables, such as smart watches and smart bands.

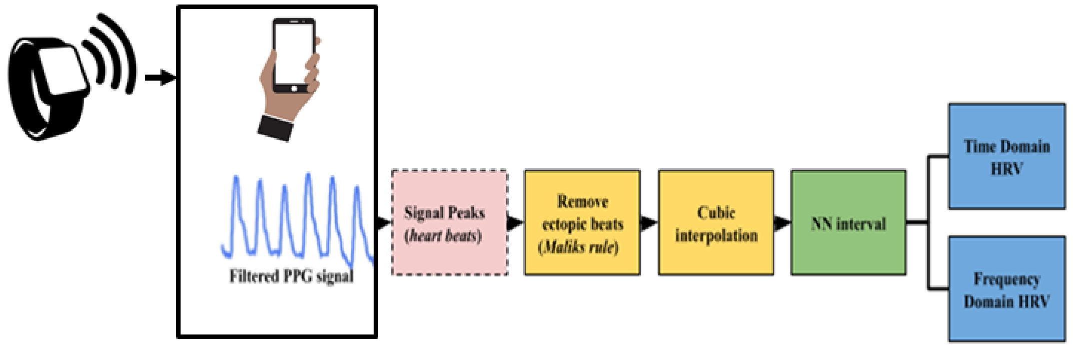


Figure 2.13: Scheme for the estimation of HRV from PPG signal.

It is important to point out that when data are noisy, a single source (i.e., HRV or EEG) may not provide information adequate enough for a reliable assessment. For progression monitoring, we have considered data and the disease condition at two time points only having an approximate two years' gap. In our opinion, longitudinal monitoring over an extended period (i.e., collecting data from a patient every night for several months) will reveal better information about disease progression and help to develop a more robust monitoring method. For future work, we have been considering optimizing and exporting the model for an android application on a smartphone for real-world deployment with human subjects and collect data every night for longitudinal monitoring over a period of a few months.

2.5 CONCLUSION

In this study, we investigated a minimalistic approach using only HRV features and time duration (%) of the SpO_2 level for the early severity estimation and continuous monitoring of OSA severity. Using only 24 ranked features from the HRV and SpO_2, an AUC of 0.91 was able to be achieved for OSA severity classification and an RMSE of 4.6 for AHI estimation. The proposed minimalistic approach can be easily integrated into edge applications and has the potential to improve healthcare monitoring significantly. Besides, it provides support for the utility of HRV or EEG features for the longitudinal monitoring of OSA patients.

REFERENCES

[1] Casola, Valentina, Aniello Castiglione, Kim-Kwang Raymond Choo, and Christian Esposito. "Healthcare-related data in the cloud: challenges and opportunities." IEEE Cloud Computing, no. 6, pp.10-14, 2014.

[2] Korosh Vatanparvar eta al., "A Generative Model for Speech Segmentation and Obfuscation for Remote Health Monitoring", The 2019 IEEE International Conference on Wearable and Implantable Body Sensor Networks, Chicago, IL, May 2019.

[3] Ion, Iulia, Niharika Sachdeva, Ponnurangam Kumaraguru, and Srdjan Čapkun. "Home is safer than the cloud! Privacy concerns for consumer cloud storage." In Proceedings of the Seventh Symposium on Usable Privacy and Security, pp. 1-20. 2011.

[4] "Mobile-Edge Computing Introductory Technical White Paper," ETSI. [Online]. Available: https://portal.etsi.org/Portals/0/TBpages/MEC/ Docs/Mobile-edge Computing - Introductory

Technical White Paper V1%2018-09-14.pdf

[5] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge Intelligence: Paving the Last Mile of Artificial Intelligence with Edge Computing," in Proceedings of the IEEE, vol. 107, no. 8, pp. 1738-1762, Aug. 2019.

[6] J. Chen and X. Ran, "Deep Learning with Edge Computing: A Review," in Proceedings of the IEEE, vol. 107, no. 8, pp. 1655-1674, Aug. 2019.

[7] B. A. Mudassar, J. H. Ko, and S. Mukhopadhyay, "Edge-cloud collaborative processing for intelligent internet of things," in Proc. the 55th Annual Design Automation Conference (DAC 2018), pp. 1–6, 2018.

[8] W. Shi, J. Cao et al., "Edge Computing: Vision and Challenges," IEEE Internet Things Journal, vol. 3, no. 5, pp. 637–646, Oct. 2016

[9] Franklin, K.A. and Lindberg, E., "Obstructive sleep apnea is a common disorder in the population—a review on the epidemiology of sleep apnea". Journal of thoracic disease, 7(8), pp.1311, 2015.

[10] Punjabi, N.M., "The epidemiology of adult obstructive sleep apnea." Proceedings of the American Thoracic Society, 5(2), pp.136-143, 2008.

[11] Kapur V, Strohl KP, Redline S, Iber C, O'Connor G, Nieto J. Underdiagnoses of sleep apnea syndrome in U.S. communities. Sleep Breath. Vol. 6, pp.49–54, 2002.

[12] Al Lawati, Nabil M., Sanjay R. Patel, and Najib T. Ayas. "Epidemiology, risk factors, and consequences of obstructive sleep apnea and short sleep duration." Progress in cardiovascular diseases, vol.51, no. 4, pp.285-293,2009.

[13] Young, T., Palta, M., Dempsey, J., Skatrud, J., Weber, S. and Badr, S., 1993. "The occurrence of sleep-disordered breathing among middle-aged adults." New England Journal of Medicine, 328(17), pp.1230-1235.

[14] A. Khandoker, C. K. Karmakar, M. Palaniswami, "Automated recognition of patients with obstructive sleep apnoea using wavelet-based features of electrocardiogram recordings", Computers in Biology and Medicine, 39 (1) pp.88–96, 2009.

[15] C. Varon, A. Caicedo, D. Testelmans, B. Buyse, S. Van Huffel, "A novel algorithm for the automatic detection of sleep apnea from single-lead ecg," IEEE Tran. on Biomedical Engineering, vol. 62, no. 9, pp. 2269-2278, Sept. 2015.

[16] D. Liu, X. Yang, G. Wang, J. Ma, Y. Liu, C. K. Peng, J. Zhang, J. Fang, “Hht based cardiopulmonary coupling analysis for sleep apnea detection”, Sleep Medicine, 13 (5), pp.503–509, 2012.

[17] H. Dickhaus, C. Maier, “Detection of sleep apnea episodes from multi-lead ecgs considering different physiological influences”, Methods of Information in Medicine, 46 (2), pp.216–221, 2007.

[18] M. Bsoul, H. Minn, L. Tamil, Apnea medassist: Real-time sleep apnea monitor using single-lead ecg, IEEE Transactions on Information Technology in Biomedicine vol. 15 no. 3, pp. 416–427, 2011.

[19] A. Zarei and B. M. Asl, "Automatic Detection of Obstructive Sleep Apnea Using Wavelet Transform and Entropy-Based Features From Single-Lead ECG Signal," in IEEE Journal of Biomedical and Health Informatics, vol. 23, no. 3, pp. 1011-1021, May 2019.

[20] Kaimakamis E, Tsara V, Bratsas C, Sichletidis L, Karvounis C, Maglaveras N (2016) “Evaluation of a Decision Support System for Obstructive Sleep Apnea with Nonlinear Analysis of Respiratory Signals”. PLoS ONE 11(3): e0150163

[21] M. J. Rahman, R. Mahajan and B. I. Morshed, "Exacerbation in Obstructive Sleep Apnea: Early Detection and Monitoring Using a Single Channel EEG with Quadratic Discriminant Analysis," 2019 9th International IEEE/EMBS Conference on Neural Engineering (NER), San Francisco, CA, USA, 2019, pp. 85-88.

[22] Kapur, V.K., Auckley, D.H., Chowdhuri, S., Kuhlmann, D.C., Mehra, R., Ramar, K. and Harrod, C.G., “Clinical practice guideline for diagnostic testing for adult obstructive sleep apnea: an American Academy of Sleep Medicine clinical practice guideline.” Journal of Clinical Sleep Medicine, 13(03), pp.479-504,2017.

[23] Epstein, Lawrence J., et al. "Clinical guideline for the evaluation, management and long-term care of obstructive sleep apnea in adults." J. of clinical sleep medicine, vol.5, no.03, pp.263-276, Jun 2009.

[24] M. Wu et al., "A New Method for Self-Estimation of the Severity of Obstructive Sleep Apnea Using Easily Available Measurements and Neural Fuzzy Evaluation System," in IEEE Journal of Biomedical and Health Informatics, vol. 21, no. 6, pp. 1524-1532, Nov. 2017.

[25] D. W. Jung, S. H. Hwang, Y. J. Lee, D. Jeong and K. S. Park, "Apnea–Hypopnea Index

Prediction Using Electrocardiogram Acquired During the Sleep-Onset Period," in IEEE Transactions on Biomedical Engineering, vol. 64, no. 2, pp. 295-301, Feb. 2017.

[26] Saha, Shumit, et al. "Apnea-hypopnea index (AHI) estimation using breathing Sounds, accelerometer and pulse oximeter." European Respiratory J. Open Research, vol.5, pp.63,2019.

[27] J. Jin and E. Sánchez-Sinencio, "A Home Sleep Apnea Screening Device With Time-Domain Signal Processing and Autonomous Scoring Capability," in IEEE Transactions on Biomedical Circuits and Systems, vol. 9, no. 1, pp. 96-104, Feb. 2015.

[28] Papini, G.B., Fonseca, P., van Gilst, M.M. et al. "Estimation of the apnea-hypopnea index in a heterogeneous sleep-disordered population using optimised cardiovascular features." Scientific Reports, vol. 9, no.17448, 2019.

[29] Grenèche, J., Sarémi, M., Erhardt, C., Hoeft, A., Eschenlauer, A., Muzet, A. and Tassi, P., "Severity of obstructive sleep apnoea/hypopnoea syndrome and subsequent waking EEG spectral power." European Respiratory Journal, 32(3), pp.705-709, 2008.

[30] "Taking an ECG with the ECG app on Apple Watch Series 4 or later" available online at https://support.apple.com/en-us/HT208955, accessed on 23 Jan, 2020.

[31] A. Benharref and M. A. Serhani, "Novel Cloud and SOA-Based Framework for E-Health Monitoring Using Wireless Biosensors," in IEEE Journal of Biomedical and Health Informatics, vol. 18, no. 1, pp. 46-55, Jan. 2014.

[32] U. Satija, et al., "Real-Time Signal Quality-Aware ECG Telemetry System for IoT-Based Health Care Monitoring," in IEEE Internet of Things Journal, vol. 4, no. 3, pp. 815-823, June 2017.

[33] G. Muhammad et al. "Smart Health Solution Integrating IoT and Cloud: A Case Study of Voice Pathology Monitoring," in IEEE Communications Magazine, vol. 55, no. 1, pp. 69-73, January 2017.

[34] Perez-Pozuelo, Ignacio, et al. "The future of sleep health: a data-driven revolution in sleep science and medicine." NPJ digital medicine 3.1 pp. 1-15, 2020.

[35] Dennis A. Dean et al., "Scaling Up Scientific Discovery in Sleep Medicine: The National Sleep Research Resource", Sleep, vol. 39, issue 5, pp. 1151–1164, May 2016.

[36] Redline S, Sanders MH, Lind BK, et al. "Methods for obtaining and analyzing

unattended polysomnography data for a multicenter study. Sleep Heart Health Research Group." Sleep, vol. 21, issue.7, pp:759–767, Nov 1998.

[37] C. Lombardi, M.F. Pengo, G. Parati, Obstructive sleep apnea syndrome and autonomic dysfunction, Autonomic Neuroscience,Volume 221, 2019, 102563, ISSN 1566-0702.

[38] Robin P. Smith, Dan Veale, Jean-Louis Pépin, Patrick A. Lévy, "Obstructive sleep apnoea and the autonomic nervous system", Sleep Medicine Reviews, vol. 2, Issue 2, pp. 69-92, 1998.

[39] Zwillich C, Devlin T, White D et al. "Bradycardia during sleep apnea. Characteristics and mechanisms." J Clin Invest, vol. 69: pp.1286-1292, 1982.

[40] J. Pan, W.J. Tompkins, "A real-time QRS detection algorithm", IEEE Trans. Biomed. Eng., vol. 32, no. 3, pp. 230-236, 1985.

[41] Malik, M., Farrell, T., Cripps, T. and Camm, A.J.,"Heart rate variability in relation to prognosis after myocardial infarction: selection of optimal processing techniques." European heart journal, vol. 10, no.12, pp.1060-1074, 1989.

[42] Goldberger AL, Amaral LAN, et al. "PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals", Circulation 101(23): e215-e220, June 2000.

[43] Suen, Colin, Clodagh M. Ryan, Talha Mubashir, Najib T. Ayas, Lusine Abrahamyan, Jean Wong, Babak Mokhlesi, and Frances Chung. "Sleep study and oximetry parameters for predicting postoperative complications in patients with OSA." Chest, vol. 155, no. 4, pp: 855-867, 2019.

[44] Wali, Siraj Omar et al. "The correlation between oxygen saturation indices and the standard obstructive sleep apnea severity." Annals of thoracic medicine vol. 15,2, pp. 70-75, 2020. doi:10.4103/ atm.ATM_215_19

[45] Dennis A. Dean, II, SpectralTrainFig , MATLAB Central File Exchange. Retrieved February 14, 2020.

[46] Guyon, I., Weston, J., Barnhill, S. and Vapnik, V., "Gene selection for cancer classification using support vector machines." Machine learning, vol. 46, no. 1-3, pp.389-422, 2002.

[47] Ambroise, Christophe, and Geoffrey J. McLachlan. "Selection bias in gene extraction on

the basis of microarray gene-expression data." Proceedings of the national academy of sciences 99.10, pp. 6562-6566, 2002.

[48] Kuhn, M., “Building predictive models in R using the caret package.” Journal of statistical software, 28(5), pp.1-26, 2008.

[49] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." Journal of machine learning research 12. Oct (2011): 2825-2830.

[50] Chollet, François. "Keras: Deep learning library for theano and tensorflow.”, URL: https://keras. io/k., 2015.

[51] Blagus, R., Lusa, L. SMOTE for high-dimensional class-imbalanced data. BMC Bioinformatics 14, 106, 2013. https://doi.org/10.1186/1471-2105-14-106

[52] N. V. Chawla, K. W. Bowyer, L. O.Hall, W. P. Kegelmeyer, “SMOTE: synthetic minority over-sampling technique,” Journal of artificial intelligence research, 321-357, 2002.

[53] Bottou, Léon. "Large-scale machine learning with stochastic gradient descent." Proceedings of COMPSTAT'2010. Physica-Verlag HD, 2010. 177-186.

[54] Neter, J., M. H. Kutner, C. J. Nachtsheim, and W. Wasserman. Applied Linear Statistical Models. 4th ed. Chicago: Irwin, 1996.

[55] Martin Fodslette Møller, “A scaled conjugate gradient algorithm for fast supervised learning”, Neural Networks, vol. 6, Issue 4, pp. 525-533, 1993.

[56] Bianchi, M.T. and Goparaju, B., “Potential underestimation of sleep apnea severity by at-home kits: rescoring in-laboratory polysomnography without sleep staging.” Journal of clinical sleep medicine, 13(4), p.551, 2017.

[57] M. J. Rahman and B. I. Morshed, "SCC Health: A Framework for Online Estimation of Disease Severity for the Smart and Connected Community," 2019 IEEE International Conference on Electro Information Technology (EIT), 2019, pp. 373-378, doi: 10.1109/EIT.2019.8834189.

Chapter 3
DEVELOPMENT OF A SMART HEALTH (SHEALTH) CENTRIC METHOD TOWARD ESTIMATION OF SLEEP DEFICIENCY SEVERITY FROM WEARABLE SENSOR DATA FUSION

3.1 INTRODUCTION

Sleep is an important biological process and plays a key role in restoring energy, solidifying and consolidating memories, and repairing body cells. It is controlled by the circadian biological clock and sleep/wake homeostasis and also helps regulate metabolism and cardiovascular function [1]. With the rise of obesity, excessive usage of personal gadgets, rapid urbanization, and other socio-economic changes, sleep/wake homeostasis is increasingly impacted, disrupting the normal circadian rhythm and healthy sleep. Good quality sleep is essential for optimal health and improved quality of life. Neither body nor the brain can function properly without sufficient sleep. Research suggests that complete sleep deprivation significantly impairs attention and working memory [2]. Moreover, it also affects other important functions, such as long-term memory and decision-making. Even partial sleep deprivation can negatively impact attention, and vigilance in the long run [3]. Moreover, sleep deficiency can lead to physical and mental health problems, injuries, loss of productivity, and even greater risk of life-threatening diseases [4-5].

Sleep health is most commonly evaluated by administering standard questionnaires, along with a sleep test where a polysomnogram is captured. The questionnaire-based approach has many limitations including high bias, long evaluation period, etc. Polysomnography is expensive, has limited availability, and is much less user-friendly. In recent years, development and use of multi-modal sensors and technologies has greatly expanded in order to monitor key aspects of sleep, such as sleep patterns, wellness applications, sleep coaching of individuals with chronic conditions, etc. [6]. For example, Kuo et al. developed an actigraphy-based wearable device for sleep quality assessment [7], and Mendonca et al. have proposed a method for sleep quality estimation using an electrocardiogram by cardiopulmonary coupling analysis [8]. Azimi et al. have described an objective, longitudinal IoT-based approach for sleep quality assessment

[9] where they were able to estimated average sleep quality average to classify sleep into good or poor quality. Additionally, Bsoul et al. have developed a Sleep Efficiency Index based on ECG features using Support Vector Machines [10]. Finally, some commercial devices (e.g., the Fitbit Charge smart band by Fitbit Inc., USA; the Apple Watch developed by Apple Inc., USA; and the Oura sleep ring produced by Oura Health Ltd., Finland) have appeared that attempt to estimate sleep scores from non-polysomnographic measures. However, most of the previous approaches have been focused on sleep quality assessment or sleep score estimation. Sleep deficiency provides a more specific evaluation of a possible sleep disorder than do sleep scores or sleep quality, as sleep deficiency includes lack of enough sleep (sleep deprivation), not obtaining all types of sleep that the human body needs, and poor quality of sleep [11]. Early detection of sleep deficiency is beneficial to avoid many linked chronic health problems, including heart disease, kidney disease, high blood pressure, diabetes, stroke, obesity, and depression. A method for objective assessment of sleep deficiency from wearable sensor data alone is not well-established and needs further investigation.

The main objective of the work reported here was to develop a physiological sensor data-based objective sleep deficiency assessment method that can be easily integrated with user-friendly wearables; e.g., smartwatch, smart band, etc. We proposed a mathematical model to facilitate a quantitative evaluation of sleep deficiency based on polysomnogram features. Then we addressed the same problem of estimating sleep deficiency severity when polysomnogram data are not available, with a machine learning-driven model using ECG/EEG based features that can be captured by wearables. The estimated sleep deficiency severity using the machine learning-based method has been validated against the ground truth established from polysomnography measurements. We hope that the results from this effort will help pave the way for automated sleep deficiency severity assessment using single-channel EEG.

3.2 MATERIALS AND METHODS

3.2.1 Smart Health (sHealth) Framework

We define “Smart Health (sHealth)” as a system that uses embedded artificial intelligence, such as edge computing, machine learning, etc., which is capable of improving healthcare via users’ smart devices, wearables, and the Internet of Things (IoT) centric solutions. This type of system would not only benefit and monitor an individual user’s health status, but it would also permit collection of spatiotemporal community-wide data for collective and social well-being and inform policy makings. It is an emerging paradigm for efficient processing, sharing, and visualization of healthcare data, which is coming from different IoT devices and wearable sensors. sHealth, thus, can be perceived as an upgraded and extended version of Mobile Health (mHealth). We previously reported a framework for sHealth and conducted a pilot study to evaluate the technical feasibility of the framework [12]. The system's architecture for the sHealth framework is shown in Fig. 3.1. The main components in the framework are various sensors, such as battery-less, body-worn passive sensors with a scanner (i.e., reader for the passive sensors), commercial wearables, a custom smartphone app (SCC-Health app), and a custom web server (SCC-Health server). Details of the design and functionality of the sensors and scanner can be found elsewhere [13]. For physiological data collection, we utilized novel inkjet-printed (IJP) sensors in addition to commercial wearables, such as a smart wristband (Mi Band 2, Xiaomi) and a fingertip pulse oximeter (CMS50E, Crucial Medical Systems) [14]. The IJP sensors were zero-power, analog, wireless, and fully passive. Data collected in the IJP sensors were pre-processed and digitized by a custom-made scanner. Data from the scanner is

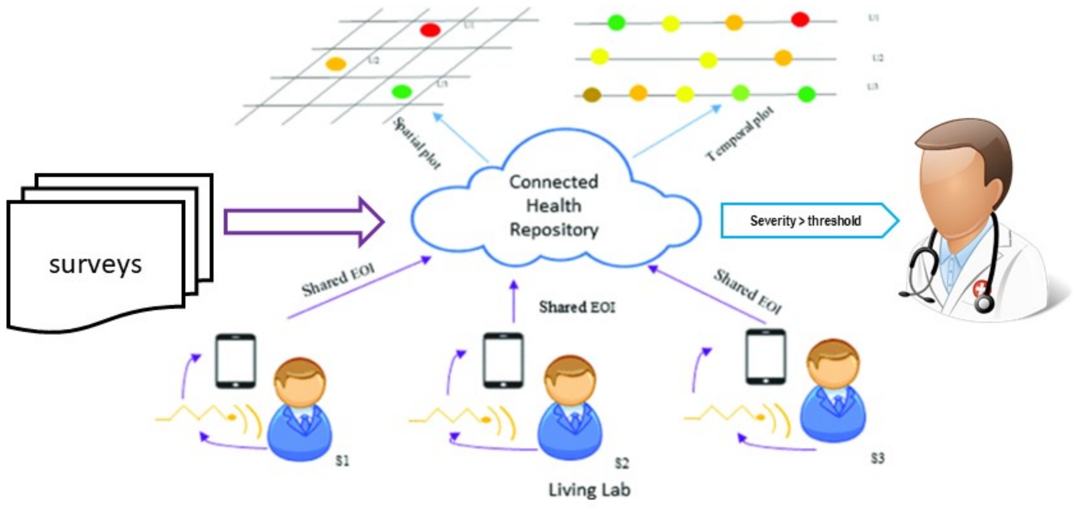


Figure 3.1: Workflow for Events of interest capture and spatiotemporal visualization in sHealth

transmitted to the smartphone via Bluetooth. Data reliability checks, feature extractions, and classification or regression analyses using pre-trained machine learning models are performed in the smartphone for disease detection and severity assessment. The computed severity of the disease is then visualized in the smartphone as well as shared with the webserver using Wi-Fi (or a cellular network) for observing the temporal and spatial distribution of the diseases. The webserver is accessible at http://sccmobilehealth.com. The app was developed using an Android studio 3.1 with build tool version 25.0.2 and the minimum SDK level of 19. The pre-trained machine learning models integrated with the app for Events of Interest (EoI) computation were developed and evaluated in WEKA. Additionally, electronic and paper-based surveys were conducted to collect user-reported symptoms and user feedback [14].

The process of spatiotemporal visualization is fully automated and near real-time. JavaScript object notation (JSON) has been used to share data from the android smartphone to the database in the webserver [15]. The shared data contains participants' anonymized user ID, area code (hashed), computed EoI, the algorithm name used for EoI computation, and timestamp of data collection. For personalized monitoring of diseases, temporal trends of disease severity for a participant can be visualized using a time plot graph [16]. A flow graph has been used for community health trend monitoring over time. In addition to that, a spatial plot was used to visualize the severity of the disease in different areas at periodic intervals (averaged) [16]. Color coding has been used to indicate severity where red indicates the highest severity and green indicates the lowest severity.

3.2.2 Sleep Health Assessment Overview

As shown in Figure 3.2, conventional methods of sleep health assessment fall under two broad categories- subjective and objective assessment of sleep. Subjective assessment of sleep deficiency using standard questionnaires is well investigated and is widely used in clinical practice. Some of the more well-accepted and popular methods for subjective sleep quality assessment are the Pittsburgh Sleep Quality Index (PSQI), the Epworth Sleepiness Scale (ESS), and the Functional Outcome of Sleep Questionnaire (FOSQ). The PSQI is a multi-component questionnaire that asks individuals to respond to an array of questions pertaining to their sleep over the prior month [17]. The following 7 components are scored from 0 (indicating no difficulty) to 3 (indicating severe difficulty): subjective sleep quality, sleep latency, sleep duration, habitual sleep efficiency, sleep disturbances, use of sleeping medication, and daytime dysfunction. Score values range from a low of 0 to a high of 21 (indicating severe difficulties in all assessed areas), with values > 5 indicating poor sleep quality in general. FOSQ has 30 questions related to activity levels, vigilance, intimacy and relationships, general productivity, and social outcomes [18]. The potential range of scores for each subscale is 1 – 4 with higher scores indicating greater insomnia severity. Similarly, in ESS the subject assigns a score of 0-3 for 8 questions aimed

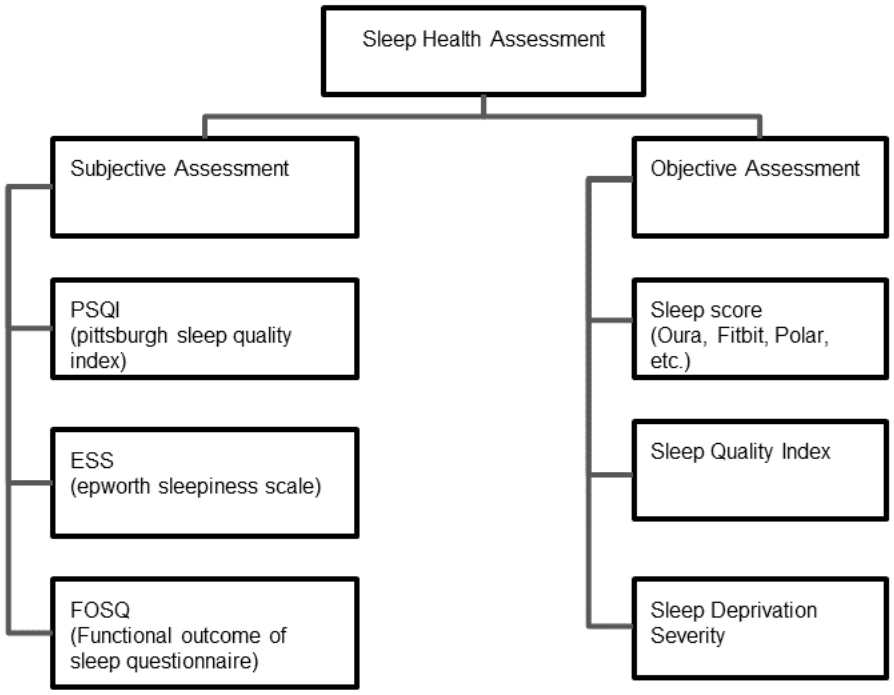

Sleep Health Assessment
Subjective Assessment
Objective Assessment
PSQI (pittsburgh sleep quality index)
ESS (epworth sleepiness scale)
FOSQ (Functional outcome of sleep questionnaire)
Sleep score (Oura, Fitbit, Polar, etc.)
Sleep Quality Index
Sleep Deprivation Severity

Fig. 3.2: Overview of subjective and objective methods for sleep health assessments

at assessing daytime sleepiness. A total score of 16-24 indicates excessive daytime sleepiness, suggesting the need for medical attention [19]. The Karolinska Sleep Diary (KSD) is another questionnaire that was developed to assess subjective sleep quality [20]. This diary contains twelve items, of which most have a scale graded from five to one.

Subjective reports of sleep quality are important in the clinical setting and can help determine whether further screening and/or treatment for a sleep complaint might be warranted [21]. However, subjective methods are prone to high bias, require active user participation, and need a longer period (2 weeks - 1 month) for fully evaluating sleep deficiency. Objective sleep quality consists not only of the total duration of sleep, but also the architecture of sleep (amount of the different sleep stages across the sleep episode), the amount of wake time during the sleep episode, and the frequency and duration of awakenings across the night [22]. Prominent quantitative metrics that are used for objective sleep assessment are the Sleep Quality/Efficiency Index and the Sleep Score. Definitions and descriptions of the currently used and proposed metric for quantitative sleep assessment is provided below:

The *Sleep Quality/Efficiency Index (SQI)* –Several quantitative metrics have been developed to measure the quality of sleep from physiological sensor data. However, a standard and well-established definition for the term 'Sleep Quality' has yet to be developed. Rather, this term is typically used to refer to a score computed from a collection of quantitative sleep measures; i.e., sleep duration, sleep onset time, degree of fragmentation, etc. [23].

The concept of a "Sleep Score" has been introduced mainly by commercial entities; i.e., Fitbit, Polar, Oura, Apple, etc. A sleep score is typically tracked by smartphone apps and is based on data collected using a smart band or a smartwatch during sleep. Fitbit computes an overall sleep score as a sum of individual scores, including sleep duration, sleep quality, and restoration, to arrive at a total score of up to a value of 100. Sleep score ranges are: Excellent: 90-100, Good: 80-89, Fair: 60-79, Poor: Less than 60, with the majority of individuals having a score between 72 and 83 [24]. A previous validation study showed that the performance of Fitbit smart bands is promising in detecting sleep-wake states and sleep stage composition relative to the gold standard polysomnogram [25]. The Oura sleep ring measures sleep using sensors that capture the body signals including resting heart rate (RHR), heart rate variability (HRV), body temperature, respiratory rate, and movement, to determine sleep patterns and compute the sleep score [26]. Sleep score-based monitoring has been criticized due to a lack of consistency in measurement and the impact of a sleep-related disease on sleep score is not well-investigated [27-28]. An ongoing scientific study is currently employing a rigorous, multi-site, multi-modality assessment of home sleep tracking technologies for diagnosing sleep disorders. [29].

Given the absence of strong supporting data, the authors have proposed a new metric—the *Sleep Deficiency Severity (SDS)*—for improving pre-clinical early evaluation of sleep deficiency that is based on a fusion of features from ECG, EEG, SpO2, and other wearable sensors. Details of the metric are described in the following sections.

3.2.3 Dataset

The Sleep Health Heart Study (SHHS) is a dataset available from the National Sleep Research Resource [30]. SHHS was implemented as a multi-center cohort study in two phases by the US National Heart Lung & Blood Institute. Unattended home polysomnograms were obtained for both the phases of SHHS by certified and trained technicians. The polysomnogram data was saved in the European Data Format (EDF). Data processing and initial scoring were accomplished using Compumedics software (Compumedics Ltd., Australia) as part of SHHS. Two manual scorings were included to annotate the database with sleep duration, sleep efficiency, arousal index, sleep stages, oxygen saturation level, etc. A dataset of 500 subjects containing good quality data for both ECG and EEG is available from the dataset provider and is recommended for use in a research study. In our study, for developing the regression models we used this dataset of 500 subjects. The gender distribution of records in the dataset is as follows: male- 231, female- 269. The age of the subjects ranges from 44 to 89 years old with a mean of 65 years old and a standard deviation of 10.41 years. The body mass index (BMI) of the subjects ranges from 18 – 46 with a mean of 27.51 kilograms per square meter and a standard deviation of 4.11 kilograms per square meter.

3.2.4 Mathematical Model for Baseline SDS Score

Guidelines for computing a composite sleep health score from polysomnographic measures have been developed and reported in previous research studies [31-32]. In this study, we developd a generalized mathematical model for computing the baseline SDS score. The model is described in equation 3.1 below, where Z_{neg} is the sleep attribute (normalized) that increases sleep

$$\lfloor Sleep\ Deficiency\ Severity \rfloor = \left\{\frac{1}{m}\sum_{i=1}^{m} Z_{neg(i)} - \frac{1}{n}\sum_{j=1}^{n} Z_{pos(j)}\right\} x\ 100 \tag{3.1}$$

Where $Z = \frac{X - min(X)}{max(X) - min\,(X)}$

deficiency (i.e., higher is responsible for more sleep deficiency), Z_{pos} is the sleep attribute (normalized) that reduces sleep deficiency (i.e., higher is better), m is the total number of negative attributes, and n is the total number of positive attributes. The positive attributes available from the SHHS dataset are as follows:

Sleep time- Duration of entire sleep.

Sleep efficiency - Percentage of time in bed that was spent sleeping, or the ratio of total sleep time to total time in bed, expressed as a percentage.

Time deep sleep (%) - Percent time in sleep stages 3 and 4.

Time REM sleep (%) - Percent Time in rapid eye movement sleep (REM).

SpO2 (%) - Average oxygen saturation (SpO2) level in sleep.

The negative attribute available from SHHS are provided by 1 measure—
the *Sleep Fragmentation Index (SFI)*, which is presented as the total number of arousals per hour of sleep; i.e., ratio of the count of arousals to total sleep time in hours. and the Sleep Deficiency Severity (SDS), wherein all of the attributes have been normalized on a scale of 0-1. The value of each negative attribute is subtracted from 1 in order to achieve a consistent "higher is better" rule. Then, the attribute values are summed up to develop a composite score. The composite score is multiplied by 100 and divided by the total number of positive and negative attributes to obtain the SDS in the range of 0-100. Statistical analyses were conducted to investigate the relationship of baseline SDS values with age, gender, and BMI. The partial correlation of for SDS (controlling for age and BMI) with HRV and EEG features was investigated, as well.

3.2.5 Machine-learning driven method for SDS estimation

We extracted features from the wearable sensor data, used Monte Carlo Feature Selection for selecting the best features, and explored machine learning and deep learning methods to develop a machine learning-based regression model, as described below:
Feature Extraction – HRV features were extracted from single channel ECG where R peaks detection was carried out using Pan-Tompkins's algorithm and RR intervals were processed followings *malik's* recommendation to rule out ectopic beats and outliers [33-34]. Similarly, spectral features of EEG were extracted from single channel EEG. Please refer to pp.22-23 for details of the feature extractions. Spectral features of EEG have been shown in Table 3.1

Table 3.1: Spectral Features Extracted from the EEG.

EEG Band	Frequency (Hz)	Features
Slow OSC	0.5 -1	Power- REM , NREM, Total
Delta	0.5 – 4	Power- REM , NREM, Total
Theta	4 – 8	Power- REM , NREM, Total
Alpha	8 - 13	Power- REM , NREM, Total
Sigma	12 – 14	Power- REM , NREM, Total
Beta	13 – 30	Power- REM , NREM, Total
Gamma	36 – 90	Power- REM , NREM, Total

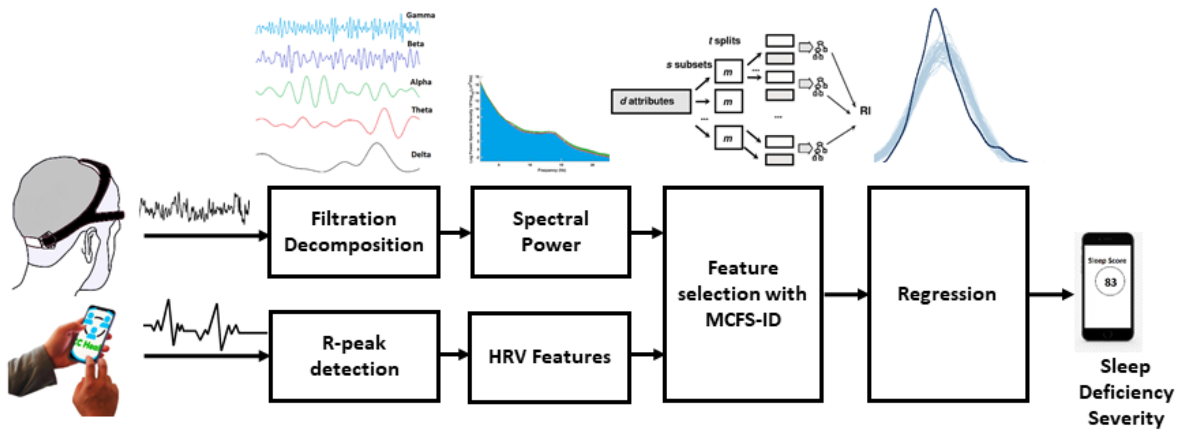


Fig. 3.3 Method for feature extraction, feature selection, and regression for sleep deficiency severity.

Monte Carlo Feature Selection and Inter-dependency Discovery- Feature selection was performed primarily to compare the relative importance of the ECG and EEG-based features for SDS estimation. Monte-Carlo Feature Selection (MCFS) and inter-dependency discovery was used for ranking feature importance. In MCFS the relative importance of features is estimated by building hundreds of trees for a randomly selected subset of features [35]. In a mathematic notion, *i* subsets of m randomly selected features are constructed where m << n, with *n* being the total number of features and for each subset, *k* trees are constructed and their performance is assessed for classification/ regression. Finally, *i* x *m* trees are constructed and evaluated. The procedure is illustrated in Figure 3.3. Weighted accuracy of a tree as defined by equation 3.2, which is used as a metric to assess the classification or regression ability of the tree.

$$Wac = \frac{1}{c}\sum_{i=1}^{c}\frac{n_{ij}}{n_{i1}+n_{i2}+\cdots..+n_{ic}}$$

Where c = number of classes, i, j = 1,2, …, c; nij is the number of samples from class i classified as class j , and $\sum n_{ij} = n$ is the number of all samples.

Equation 3.2

The Relative Importance (RI) of feature g_d denoted by RI_{gd} is defined by equation 3.3.

$$RI_{gd} = \sum_{x=1}^{m.k} W{ac_x}^{u} \sum_{r_{gd(x)}} IG\left(r_{gd}(x)\right)\left(\frac{no.\ in\, r_{gd}(x)}{no.\ in\, x}\right)^{v}$$

Where Wac stands for the weighted accuracy for x^th^ tree; $IG\left(r_{gd}(x)\right)$ stands for the Information Gain for node $r_{gd}(x)$; (no. in $r_{gd}(x)$) denotes the number of samples in the node $r_{gd}(x)$; (no. in x) denotes the number of samples in the root of the xth tree; and u and v are fixed positive reals. Information Gain (IG) is measured by the Gini Index or Gain Ratio [36].

Equation 3.3

The ECG and EEG both have correlated features that introduce the problem of multi-collinearity. To deal with this, the inter-dependency discovery was used to remove features with strong pairwise interactions. The *rmcfs* package from R was implemented for feature ranking using the Monte-Carlo Feature Selection and Interdependency Discovery (MCFS-ID) methods [36]. The steps of preprocessing, feature extraction, feature selection, and regression were those as previously shown in Figure 3.3.

Regression Model- For developing the regression model, we investigated the Bayesian Regression method as well as Artificial Neural Network (ANN) approach. Bayesian inference facilitates overcoming insufficient data or poorly distributed data as it allows one to input prior values for the coefficients and the noise so that in the absence of data, the priors can take over. In a Bayesian framework, the regression model is stated in a probabilistic manner where the Bayesian sampling algorithm returns a probability distribution (known as the posterior of the effect) that is compatible with the observed data instead of a point estimate. The posterior distribution is obtained by the product of the prior distribution and the likelihood function. The model for Bayesian Linear Regression is represented in equation 3.4.

$$y \sim N(W^T X, \sigma^2 I)$$

Where response data points **y** is sampled from a multivariate Gaussian distribution that has a mean equal to the product of **W** coefficients and the predictors **X** and variance of $\boldsymbol{\sigma^2}$. **I** is the N X N Identity matrix [37].

Equation 3.4.

In this work, we used the Markov Chain Monte Carlo Method (MCMC) sampling and weakly informative prior for Bayesian regression. To verify convergence, the potential scale reduction statistic R-hat was used [38].

An Artificial Neural Network (ANN) is capable of approximating any linear or non-linear relationship, including multi-dimensional regression mapping problems, quite well. However, the ANN must have enough neurons in the hidden layers and the data distribution should be consistent. During the training process, an ANN fits a function on a set of inputs to produce a set of associated outputs. Once training is finished the network forms a generalization of the input-output relationship and can be utilized to generate outputs for unseen inputs. The structure of ANN has multiple layers with interconnected artificial neurons as the building blocks for each layer. Each neuron has weights that are adjusted during the training process. Training stops when any of these conditions occur: the maximum number of epochs (repetitions) is reached, the maximum amount of time is exceeded, performance is minimized to the goal, or the performance gradient falls below a minimum gradient. The ANN used in this study was a feed-forward type that had 3 layers- input, output, and hidden layer. The number of neurons in each layer is input-117, hidden- 10, output-1. The used activation functions are- *relu* for the hidden layer and *softmax* for the output layer. Levenberg-Marquardt optimization with backpropagation was used as the training algorithm [39]. The hyperparameters used for the ANN are as follows: *max epochs* = 1000, *min gradient* = 1e-7, momentum (*Mu)* = 0.001, *Mu decrease ratio* = 0.1, *Mu increase ratio* = 0.1. To facilitate proper training and evaluation the input data was randomly divided into training (80%) and test (20%) sets. Root Mean Squared Error (RMSE) and R-squared (R2) values were used for performance evaluation of both the Bayesian model and ANN. Additionally, a Pareto smoothed importance sampling (PSIS) diagnostic plot was used for the Bayesian model. Good Pareto k estimates ($k < 0.5$) in the PSIS diagnostic plot show that the model fits the data. The version of PSIS used in this work corresponds to the algorithm presented in Vehtari, Simpson, Gelman, Yao, and Gabry [40].

Assessment of Obstructive Sleep Apnea Impact- It is well established that Obstructive Sleep Apnea (OSA) has a negative consequence on sleep and is a reason for sleep deficiency. OSA induces behavioral sleep problems and bedtime resistance, which result in a significantly shortened sleep duration [41]. Apnea-hypopnea Index (AHI) is used to quantify the degree of OSA. We investigated the correlation of ESS, FOSQ, and SDS with AHI to examine which measure better captures the impact of OSA on sleep deficiency.

3.3 RESULTS

The probability density plot of SDS was computed using equation 1, which has been shown previously in Figure 3.4. The histogram of SDS follows a Gaussian distribution with a mean of 60 (N=500) and a standard deviation of 22. A boxplot comparison between the sleep deficiency severities of males and females is shown in Figure 3.5. No significant (p-value>0.05) difference was observed between the average SDS of males with that of females. SDS showed a moderate (r=-0.35, p=0.01) correlation with age; i.e., higher the age, the higher the sleep deficiency. The scatterplot of age and SDS with a trend line is visualized in Figure 3.6. Additionally, SDS showed a weak (r=-0.21, p=0.01) positive correlation with Body Mass Index (BMI). Boxplots of SDS for normal and overweight categories are shown in Figure 3.7. The overweight category had

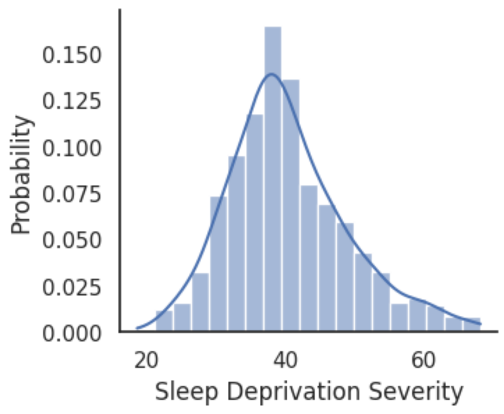


Fig. 3.4 Probability density plot for SDS distribution

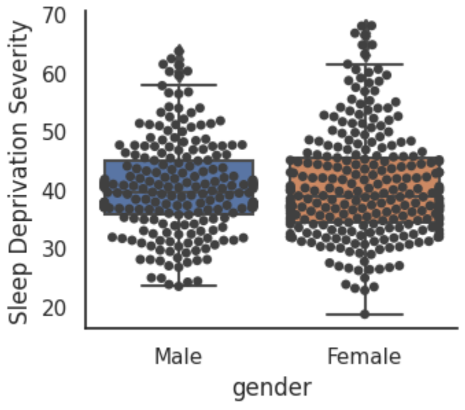


Fig. 3.5 Comparison of SDS between males and females

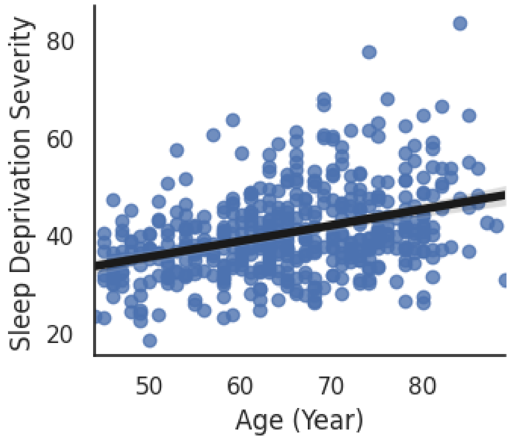


Fig. 3.6 Correlation of SDS with Age

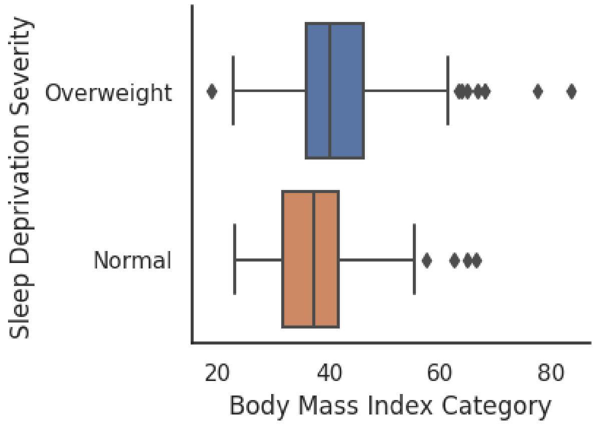


Fig. 3.7 SDS for BMI Categories

a higher SDS. The partial correlation (controlling for age and BMI) of HRV and EEG features with baseline SDS, computed using equation 1 indicated a significant correlation for several features. A co-variate analysis was performed to investigate the relationship of SDS with these features when controlled for age and BMI. The best 5 features from each sensor showing a significant correlation with SDS are listed in Table 3.2.

Table 3.2: Partial Correlation of SDS with HRV and EEG features.

HRV Features			**EEG Features**		
Feature	r	p-value	Feature	r	p-value
AVNN	0.09	0.04	slowosc_nrem	0.39	0.01
pNN10	0.08	0.04	delta_nrem	0.39	0.01
HR	-0.11	0.02	slowosc_sleep	0.33	0.01
VLF	-0.11	0.02	delta_sleep	0.29	0.01
LF/HF	-0.13	0.01	theta_rem	0.25	0.01

* Variables in the table are described in Table 2.2-2.4

Although both the HRV and EEG features revealed significant partial correlations with SDS, the correlation for the EEG features was much stronger than that for the HRV features, indicating that the EEG features had relatively higher importance than the HRV features in estimating SDS. Hence, in developing the regression method, only the EEG and anthropometric measures were used.

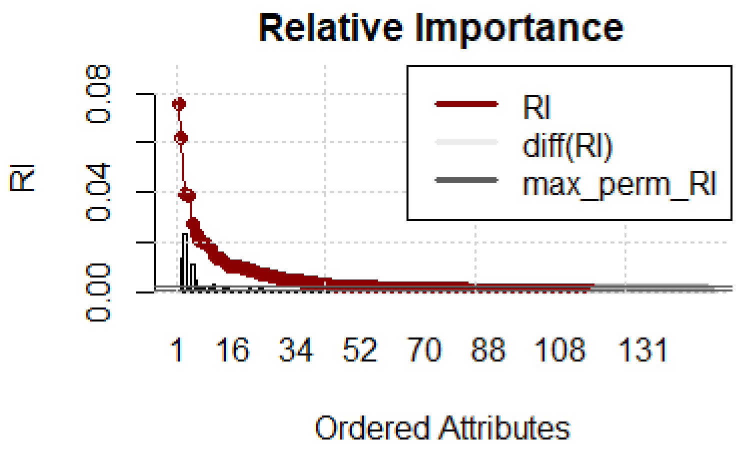


Figure 3.8: Relative importance of features.

The MCFS results indicate that out of 150 features 117 were important based on the cut-off value of feature relative importance (RI) as shown in Figure 3.8. The line with the red/gray dots provides the RI values, the vertical bar plot displays the difference δ between consecutive RI values. Informative features are separated from non-informative ones by the cutoff value and are presented in the plot as red and gray dots, respectively. The convergence of the MCFS-ID algorithm is shown in Figure 3.9. The distance function (red line) shows the difference between two consecutive rankings, where zero means no changes between two rankings (see the left y-axis). The common part (colored in blue) indicates the fraction of features that overlap for two different rankings (see the right y-axis). The ranking stabilizes after some iterations: the distance tends toward zero and the common part tends toward 1. Beta1 shows the slope of the tangent of a smoothed distance function. If beta1 tends to 0 (the right y-axis) then the distance is displayed as a flat line. The top-ranked 20 features based on the normalized relative importance by MCFS-ID are shown in Figure 3.10.

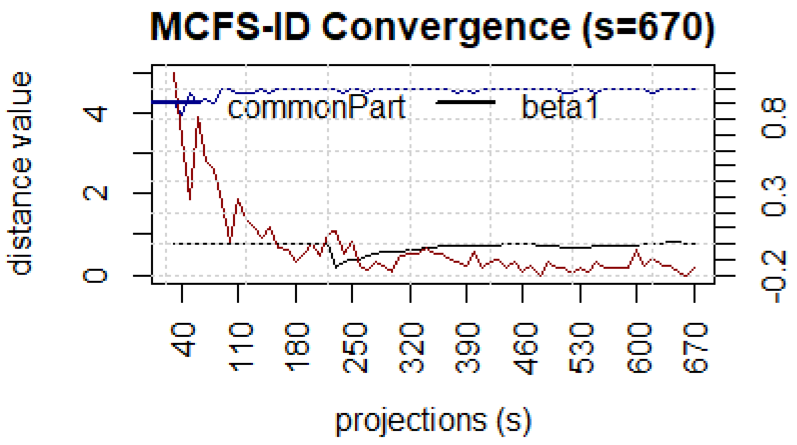


Figure 3.9: Convergence of Monte-Carlo feature selection.

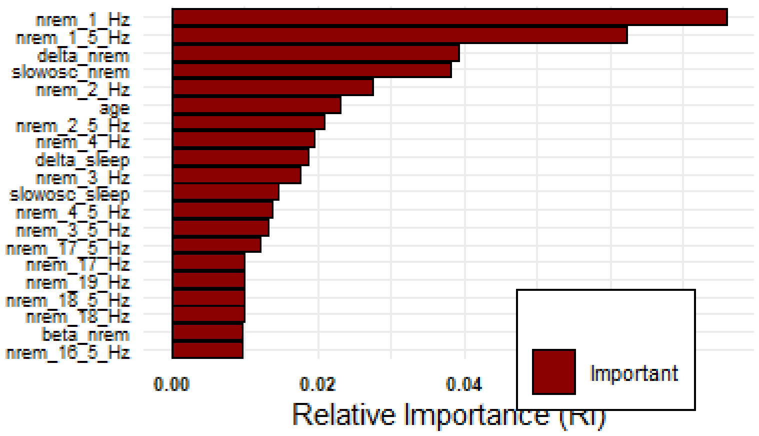


Figure 3.10: Top 20 features by MCFS-ID algorithm.

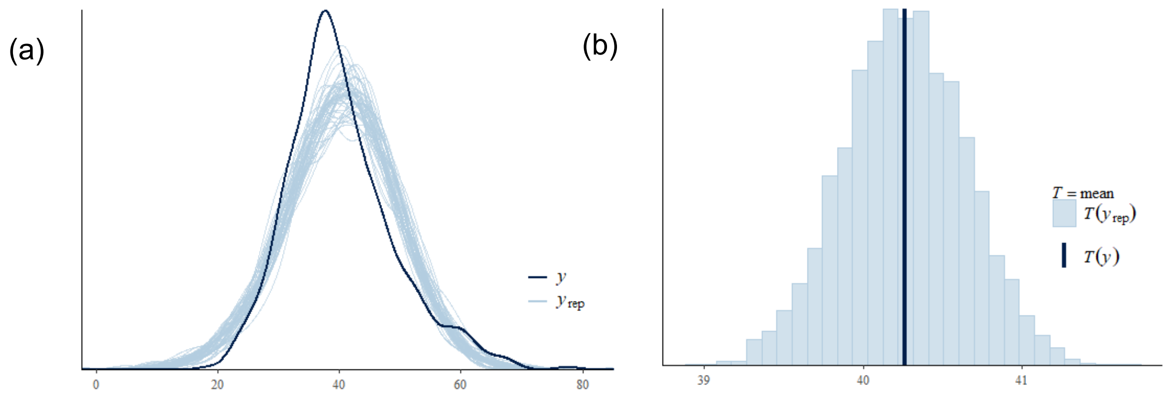


Figure 3.11: (a) posterior predictive check on MCMC sampler; (b) density plot of Bayesian model estimated SDS including the point estimate.

The distribution of posterior R2 for estimating SDS using Bayesian regression indicates an approximately normally distributed pattern. In MCMC diagnostics R-hat values for all parameters were less than 1.1. A posterior predictive check on the MCMC sampler is shown in Figure 3.11(a). The dark blue line shows the observed data, while the light blue lines are simulations from the posterior predictive distribution. The patterns for both distributions agree, with some deviations for the peak. Figure 3.11(b) shows the probability density plot for the estimated SDS using Bayesian regression, where the solid line indicates the point estimate from the Ordinary least squares method. The plot–of–fit for the Bayesian regression model is shown in Figure 3.12, where the R-squared value = 0.60 and RMSE = 5.63. The PSIS diagnostics plot for the Bayesian model is shown in Figure 3.13, which reveals that only a few points are outside of the acceptable threshold. The estimated shape parameter *k* for each observation is used as a measure of the observation's influence on the posterior distribution of the model.

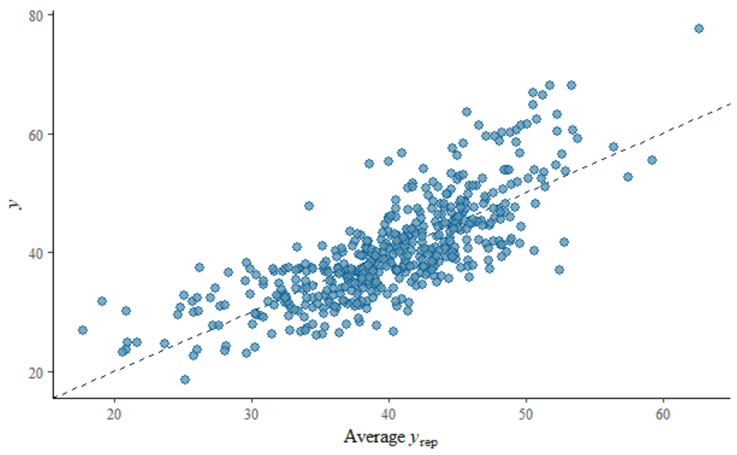


Figure 3.12: Plot–of–fit for the Bayesian regression model.

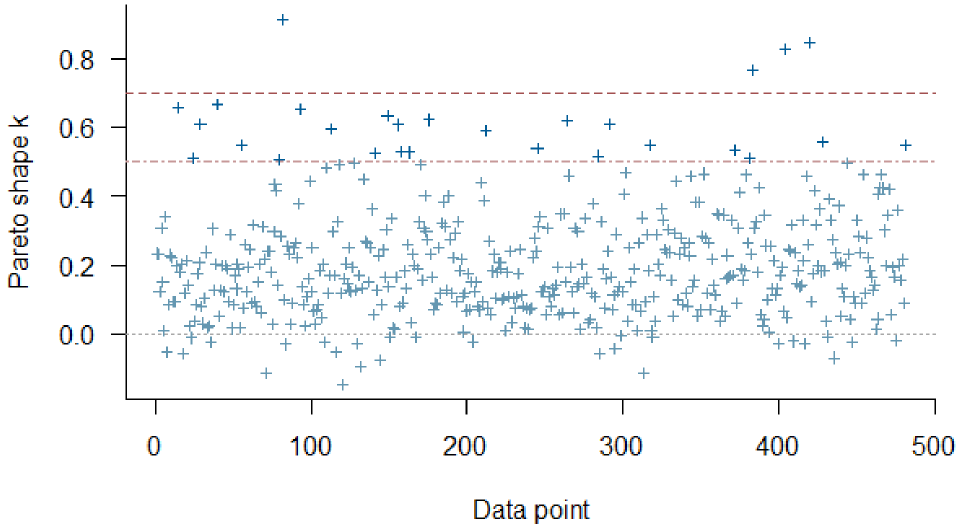


Figure 3.13: PSIS diagnostic plot and regression plot for Bayesian method.

For SDS estimation, ANN achieved a performance of RMSE of 4.65 and R-squared value of 0.86 in the training set, and an RMSE of 5.47 and R-squared value of 0.67 in the test set. The fit of the regression plot for the ANN model is shown in Figure 3.14. The dashed line indicates the ideal trend line and the solid line indicates the fitted trend line for the actual versus predicted values. The histogram of prediction error showed symmetrically skewed and almost normally distributed patterns with a higher frequency in the error bin ± 2. The residual plot for the regression analysis shows a random scattering around the zero lines.

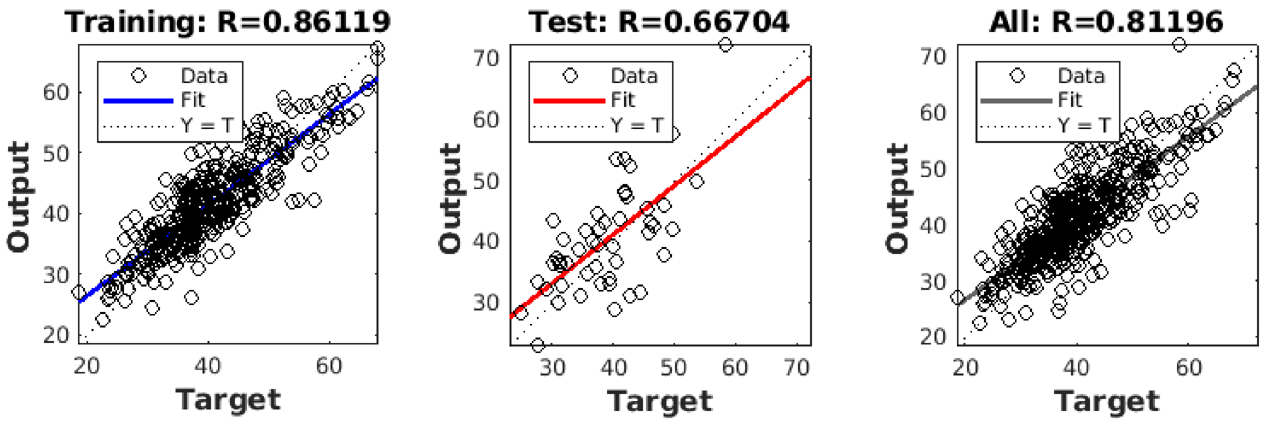


Figure 3.14: Performance of the Regression model for the train, validation, and test sets.

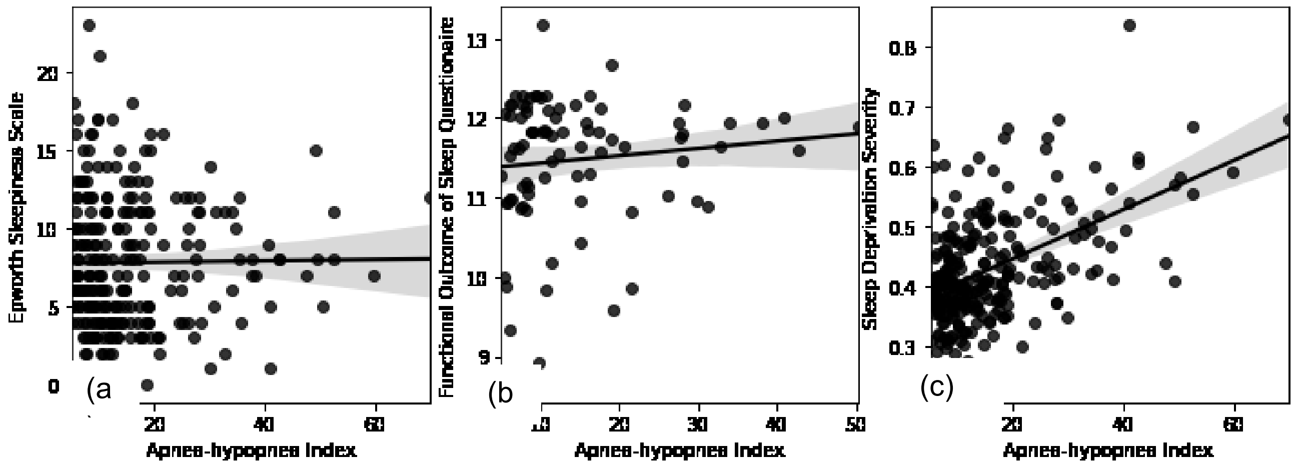


Figure 3.15: Correlations of (a) ESS with AHI, (b) FOSQ with AHI, and (c) SDS with AHI.

Figure 3.15(a) shows the impact of OSA as captured by the Epworth Sleepiness Scale (ESS). As shown, the ESS did not reveal an informative trend and failed ($r < 0.1$) to capture the impact of severe OSA on the sleep deficiency measure of OSA patients. Similarly, Figure 3.15(b) shows the correlation of the Functional Outcome of Sleep Questionnaire (FOSQ) with the apnea-hypopnea index. The trend in this case also failed ($r < 0.19$) to capture the impact of OSA severity on sleep deficiency. Figure 3.15(c) shows that SDS, as computed using the proposed method, shows a modest positive correlation ($r = 0.31$) with AHI. As OSA severity increased, SDS also proportionately increased.

3.4. DISCUSSION

While quantification and longitudinal monitoring of sleep deficiency are beneficial for early diagnosis and continuous monitoring of the presence of a sleep disorder may facilitate corrective habitual actions and practices that adversely affect good sleep, it is noteworthy that sleep deficiency was linked not only with the physiological disorder but also with emotional stress and other factors. In order to reduce the variability in everyday measurement, a moving average over a week or longer period, as well as sleep pattern visualization, may provide better insights when added to the SDS score. Signal quality and data reliability also impact the measurements and, hence, a data reliability metric may be helpful for enhancing the usability of the method. Moreover, it is noteworthy that we could not directly compare the utility of SDS with that of the sleep score as the sleep score formulae used by commercial entities are not publicly available to the best of our knowledge.

5. CONCLUSIONS

In this study, we analyzed SDS and its relationship with HRV and EEG-based features. Feature ranking, using MCFS-ID, was implemented for identifying the most informative features for SDS estimation. Finally, we developed a regression method using ANN for SDS score estimation from spectral features of a single-channel EEG. The findings from this study increased the interpretability of SDS and helps to pave the way for using SDS as a potential indicator for automated sleep disorder checks using wearables. In future studies, we are aiming a large scale deployment of the model for longitudinal monitoring of SDS with wearables.

REFERENCES

1. Mukherjee, Sutapa, Sanjay R. Patel, Stefanos N. Kales, Najib T. Ayas, Kingman P. Strohl, David Gozal, and Atul Malhotra. "An official American Thoracic Society statement: the importance of healthy sleep. Recommendations and future priorities.", American journal of respiratory and critical care medicine, vol.191, no. 12, pp: 1450-1458, 2015.
2. Dai, Cimin, Ying Zhang, Xiaoping Cai, Ziyi Peng, Liwei Zhang, Yongcong Shao, and Cuifeng Wang. "Effects of sleep deprivation on working memory: change in functional connectivity between the dorsal attention, default mode, and fronto-parietal networks." Frontiers in Human Neuroscience 14 (2020).
3. Alhola, Paula, and Päivi Polo-Kantola. "Sleep deprivation: Impact on cognitive performance." Neuropsychiatric disease and treatment vol. 3,5 (2007): 553-67.
4. Spiegel, Karine, Esra Tasali, Rachel Leproult, and Eve Van Cauter. "Effects of poor and short sleep on glucose metabolism and obesity risk.", Nature Reviews Endocrinology, vol. 5, no. 5, pp: 253, 2009.
5. Sharma, Monika, J. P. S. Sawhney, and Samhita Panda. "Sleep quality and duration–Potentially modifiable risk factors for Coronary Artery Disease?.", indian heart journal, vol. 66, no. 6, pp.: 565-568, 2014.
6. Perez-Pozuelo, I., Zhai, B., Palotti, J. et al. The future of sleep health: a data-driven revolution in sleep science and medicine. npj Digit. Med. 3, 42 (2020). https://doi.org/10.1038/s41746-020-0244-4.
7. C. Kuo, Y. Liu, D. Chang, C. Young, F. Shaw and S. Liang, "Development and Evaluation of a Wearable Device for Sleep Quality Assessment," in IEEE Transactions on Biomedical Engineering, vol. 64, no. 7, pp. 1547-1557, July 2017.
8. F. Mendonça, S. S. Mostafa, F. Morgado-Dias and A. G. Ravelo-García, "Sleep Quality Estimation by Cardiopulmonary Coupling Analysis," in IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 26, no. 12, pp. 2233-2239, Dec. 2018, doi: 10.1109/TNSRE.2018.2881361.
9. I. Azimi et al., "Personalized Maternal Sleep Quality Assessment: An Objective IoT-based Longitudinal Study," in IEEE Access, vol. 7, pp. 93433-93447, 2019, doi: 10.1109/ACCESS.2019.2927781.

10. Bsoul, Majdi, Hlaing Minn, Mehrdad Nourani, Gopal Gupta, and Lakshman Tamil. "Real-time sleep quality assessment using single-lead ECG and multi-stage SVM classifier." In 2010 Annual International Conference of the IEEE Engineering in Medicine and Biology, pp. 1178-1181. IEEE, 2010.
11. Sleep Deprivation and Deficiency, https://www.nhlbi.nih.gov/health-topics/sleep-deprivation-and-deficiency, National Institute of Health, US Dept. of Health and Human Services, retrieved on July 4, 2021.
12. M. J. Rahman and B. I. Morshed, "SCC Health: A Framework for Online Estimation of Disease Severity for the Smart and Connected Community," 2019 IEEE International Conference on Electro Information Technology (EIT), 2019, pp. 373-378, doi: 10.1109/EIT.2019.8834189.
13. Zaman, Md Sabbir, and Bashir I. Morshed. "A low-power portable scanner for body-worn Wireless Resistive Analog Passive (WRAP) sensors for mHealth applications." *Measurement* 177 (2021): 109214.
14. M. J. Rahman, B. I. Morshed and B. Harmon, "A Field Study to Capture Events of Interest (EoI) from Living Labs Using Wearables for Spatiotemporal Monitoring Towards a Framework of Smart Health (sHealth)," 2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), 2020, pp. 5943-5947, doi: 10.1109/EMBC44109.2020.9175771.
15. Morshed, Bashir I., Brook Harmon, Md Sabbir Zaman, Md Juber Rahman, Sharmin Afroz, and Mamunur Rahman. "Inkjet printed fully-passive body-worn wireless sensors for smart and connected community (SCC)." *Journal of Low Power Electronics and Applications* 7, no. 4 (2017): 26.
16. Afroz, Sharmin, and Bashir I. Morshed. "Web Visualization of Temporal and Spatial Health Data from Smartphone App in Smart and Connected Community (SCC)." In *2018 IEEE International Smart Cities Conference (ISC2)*, pp. 1-6. IEEE, 2018.
17. Buysse, Daniel J., Charles F. Reynolds III, Timothy H. Monk, Susan R. Berman, and David J. Kupfer. "The Pittsburgh Sleep Quality Index: a new instrument for psychiatric practice and research." Psychiatry research, vol. 28, no. 2, pp: 193-213, 1989.

18. Weaver, Terri E., Andréa M. Laizner, Lois K. Evans, Greg Maislin, Deepak K. Chugh, Kerry Lyon, Philip L. Smith *et al.*"An instrument to measure functional status outcomes for disorders of excessive sleepiness." Sleep, vol. 20, no. 10, pp: 835-843, 1997.
19. Johns, Murray W. "A new method for measuring daytime sleepiness: the Epworth sleepiness scale." Sleep, vol. 14, no. 6, pp: 540-545, 1991.
20. Åkerstedt, Torbjörn, K. E. N. Hume, David Minors, and J. I. M. Waterhouse. "The subjective meaning of good sleep, an intraindividual approach using the Karolinska Sleep Diary." *Perceptual and motor skills* 79, no. 1 (1994): 287-296.
21. O'donnell, Deirdre, Edward J. Silva, Mirjam Münch, Joseph M. Ronda, Wei Wang, and Jeanne F. Duffy. "Comparison of subjective and objective assessments of sleep in healthy older subjects without sleep complaints." *Journal of sleep research* 18, no. 2 (2009): 254-263.
22. Zhang, Lin, and Zhong-Xin Zhao. "Objective and subjective measures for sleep disorders." Neuroscience Bulletin 23, no. 4 (2007): 236-240. A. D. Krystal and J. D. Edinger, "Measuring sleep quality", Sleep Med., vol. 9, pp. S10-S17, Sep. 2008.
23. Sun, Shuyu, Xianchao Zhao, Jiafeng Ren, Jinxiang Cheng, Junying Zhou, and Changjun Su. "Characteristics of objective sleep and its related risk factors among Parkinson's disease patients with and without restless legs syndrome." Frontiers in Neurology 12 (2021).
24. What's sleep score in the Fitbit app? https://help.fitbit.com/articles/en_US/Help_article/2439.htm. Retrieved on June 25, 2021.
25. de Zambotti, Massimiliano, *et al.* "A validation study of Fitbit Charge 2™ compared with polysomnography in adults." Chronobiology international 35.4 (2018): 465-476.
26. How Oura Measures Your Sleep, Retrieved on June 25, 2021 from https://ouraring.com/blog/sleep-score/
27. That Sleep Tracker Could Make Your Insomnia Worse, New York Times. Retrieved on June 25, 2021 from https://www.nytimes.com/2019/06/13/health/sleep-tracker-insomnia-orthosomnia.html. A version of this article appears in print on June 17, 2019, Section B, Page 3 of the New York edition of with the headline: Sleep Trackers Could Make Your Insomnia Worse

28. Liang, Zilu, and Mario Alberto Chapa Martell. "Validity of consumer activity wristbands and wearable EEG for measuring overall sleep parameters and sleep structure in free-living conditions." Journal of Healthcare Informatics Research 2.1 (2018): 152-178.
29. Toedebusch, Cristina D et al. "Multi-Modal Home Sleep Monitoring in Older Adults." Journal of visualized experiments : JoVE ,143 10.3791/58823. 26 Jan. 2019, doi:10.3791/58823
30. Quan, Stuart F., Barbara V. Howard, Conrad Iber, James P. Kiley, F. Javier Nieto, George T. O'Connor, David M. Rapoport *et al.*"The sleep heart health study: design, rationale, and methods." Sleep 20, no. 12 (1997): 1077-1085.
31. Rosipal, Roman, Achim Lewandowski, and Georg Dorffner. "In search of objective components for sleep quality indexing in normal sleep." Biological psychology 94, no. 1 (2013): 210-220.
32. Landry, Glenn J et al. "Measuring sleep quality in older adults: a comparison using subjective and objective methods." Frontiers in aging neuroscience vol. 7 166. 7 Sep. 2015, doi:10.3389/fnagi.2015.00166
33. Malik, M., Farrell, T., Cripps, T. and Camm, A.J.,"Heart rate variability in relation to prognosis after myocardial infarction: selection of optimal processing techniques." European heart journal, vol. 10 , no.12, pp.1060-1074, 1989.
34. Goldberger, Ary L., Luis AN Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and H. Eugene Stanley. "PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals." circulation 101, no. 23 (2000): e215-e220.
35. Dramiński, Michał, Alvaro Rada-Iglesias, Stefan Enroth, Claes Wadelius, Jacek Koronacki, and Jan Komorowski. "Monte Carlo feature selection for supervised classification." Bioinformatics 24, no. 1, pp.: 110-117, 2008.
36. Dramiński, Michał, and Jacek Koronacki. "rmcfs: an R package for Monte Carlo feature selection and interdependency discovery." Journal of Statistical Software 85, no. 1, pp.: 1-28, 2008.

37. Bishop, Christopher M., and Michael E. Tipping. "Bayesian regression and classification." Nato Science Series sub Series III Computer And Systems Sciences 190 (2003): 267-288.

38. Goodrich B, Gabry J, Ali I, Brilleman S (2020). “rstanarm: Bayesian applied regression modeling via Stan.” R package version 2.21.1, https://mc-stan.org/rstanarm.

39. Marquardt, Donald W. "An algorithm for least-squares estimation of nonlinear parameters." Journal of the society for Industrial and Applied Mathematics 11, no. 2, pp.: 431-441, 1963.

40. Vehtari, Aki, Daniel Simpson, Andrew Gelman, Yuling Yao, and Jonah Gabry. "Pareto smoothed importance sampling." arXiv preprint arXiv:1507.02646, 2015.

41. Owens J, Opipari L, Nobile C, Spirito A. Sleep and daytime behavior in children with obstructive sleep apnea and behavioral sleep disorders. Pediatrics. 1998 Nov;102(5):1178-84. doi: 10.1542/peds.102.5.1178. PMID: 9794951.

Chapter 4
A PILOT STUDY TOWARDS A SMART-HEALTH FRAMEWORK TO COLLECT AND ANALYZE BIOMARKERS WITH LOW-COST AND FLEXIBLE WEARABLES AT A SMART AND CONNECTED COMMUNITY

4.1 INTRODUCTION

Delayed diagnosis of disease often necessitates complicated treatment and leads to a lessened quality of life for the patient, thereby greatly increasing the healthcare burden as well as associated costs. One of the prominent ways to improve healthcare delivery and treatment outcomes is to detect a disease at the earliest stage possible. Modern technology has contributed substantially to the advancement of healthcare in a clinical setting with the introduction of precision diagnostic tools, minimally invasive surgical procedures, remote consultation, technology-based therapeutic tools, point of care delivery, etc. However, early detection of disease in a pre-clinical setting at a living lab environment is yet to be realized. Preliminary estimation of disease severity provides a much valuable insight regarding the stage of the disease and can serve to motivate a patient to seek medical attention immediately. For many diseases, the introduction of pathological conditions is reflected in the vital signs; i.e., heart rate, respiration rate, oxygen saturation level, temperature, and blood pressure [1]. In addition, biosignals, such as the Electrocardiogram (ECG) Electroencephalography (EEG), reveal a great amount of information that helps in early diagnosis. Although some of these sensing modalities might be present in smartphones, they are not usable for continuous passive monitoring. Progress in the field of wearable body sensor networks (BSN), which typically involves collection of multiple tiny wireless sensors, has made the monitoring of vital signs and capturing of biopotentials activity more reliable and user friendly.

We define “Smart Health (sHealth)” as an approach that aims to deliver improved healthcare via users’ smart devices, wearables, and Internet of Things (IoT) centric solutions that incorporates embedded artificial intelligence, such as edge computing machine learning models, that not only benefit and inform individual users, but also collects spatiotemporal community-wide data for collective and social well-being and informed policy making [2]. It is an emerging paradigm for efficient processing, sharing, and visualization of healthcare data, which is coming from different IoT devices. sHealth can be perceived as an upgraded and extended version of mHealth. mHealth has been defined as “medical and public health practice supported by mobile devices, such as mobile phones, patient monitoring devices, personal digital assistants, and other wireless devices” [3]. sHealth aims to incorporate the smart home and smart city infrastructure, related IoT devices, and other wearables for disease diagnosis, monitoring and healthcare delivery [4]. Although mHealth has already brought about revolutionary improvements and benefits for ubiquitous and pervasive healthcare delivery, it has focused mainly on personal health monitoring, whereas sHealth acknowledges the role of home, family, and community as important contributors to individual health and wellbeing, and aims to connect data, people and systems to enable long-term care rather than provide sporadic treatment to acute conditions [5]. S. Rani et al. investigated the use of Smart Health for controlling the chikungunya virus, a mosquito instinctive disease that spreads hurriedly in various parts of the country [6]. They presented an IoT-enabled model that addressed data collection from the sensors, objects, and people and gathered all of the data at the cloud to enable healthcare professionals to take preventive and control measures. R. K. Pathinarupothi, et al. reported an IoT-based smart edge system for remote health monitoring, in which wearable vital sensors transmitted data into the IoT smart edge [7]. The IoT smart edge then employed a risk-stratified protocol to trigger rapid push of alerts and personal health motifs to the physicians, and also facilitated the pull of detailed data-on-demand through the cloud. W. N. Ismail et al. proposed a convolutional neural network-based health model for regular health factors analysis in the Internet-of-Medical Things environment [8]. Their method uses the health conditions and lifestyle patterns related to chronic diseases collected through IoT-devices.

While cloud computing has its benefits with increased computational capacity, storage facility, and high reliability, thus making it suitable for unstructured big data landscape, it also has drawbacks and limitations as data transmission to the cloud imposes a huge burden on network bandwidth and the addition of millions of IoT devices in the near future may overload the computational capacity of cloud infrastructures [9]. Fog computing, where computation tasks are performed close to the user and terminal IoT devices, offers an alternative to cloud computing [10-11]. While Fog-computing helps to reduce the network bandwidth requirement and computational burden of the cloud, thus enabling a faster response, it does have maintenance, privacy, and security issues, as raw data goes out of the personal devices. Edge-computing with on-device data processing and machine learning offers a promising solution to reduce privacy and security related issues, enable real-time disease monitoring and detection, and greatly reduce network dependency [12-13]. Hence, edge computing holds tremendous potential for the Smart and Connected Health. However, edge computing is challenging because edge devices are resource constrained, as they are equipped with low processing capacity, small battery power, and limited storage [14]. Artificial Intelligence (AI) based models integrated into an edge application for automated disease detection needs to be quantized which often leads to poor performance. Minimizing the tradeoff between runtime and accuracy is challenging. A hybrid edge-cloud model may offer a better solution combining both the advantages of edge computing and cloud computing [15]. Smartphones with increased processing speed, storage capacity, and integrated sensors offer tremendous promise as an edge device in physiological sensing, monitoring, and the development of a Smart and Connected Community (SCC). However, the current sensor modalities of smartphones are very limited when attempting to record clinically relevant physiological information. Smart bands, smartwatches, and Inkjet-Printed (IJP) sensors are becoming increasingly popular day by day for physiological sensing. Although the utility of smartwatches or smart bands is well investigated for physical activity and heart rate monitoring, reliable extraction and monitoring of more informative biomarkers (e.g., heart rate variability (HRV and core-body temperature) is yet to be realized [16-19].

Production of lightweight, unobtrusive, low-cost, low-power, versatile and stretchable user-friendly sensors and documentation of their feasibility for pervasive sensing is a priority. Inkjet printing, which combines chemistry and technology for advanced manufacturing, is rising at the forefront of biosensor fabrication technologies [20]. However, when moving from theoretical understanding to realistic implementation, many technical aspects have to be considered. In this yearlong field study, we collected the nocturnal Instantaneous Heart Rate (IHR) signal from participants using wrist-worn commercial smart bands and extracted HRV features. In addition, we measured core body temperature using our custom-designed flexible IJP temperature sensor and SpO_2 with a finger pulse oximeter. Core body temperature, along with user-reported symptoms, have been used successfully for automated spatiotemporal monitoring of flu symptoms severity. The entire study was conducted in a living lab environment where participants were busy with daily life activities, which gives this study an edge over those conducted in more controlled and artificial settings.

The main focus of this study was to evaluate the feasibility of using IJP sensors for various biomedical applications, including development of design protocols, wireless data transfer, on-device data processing, assessment of user interaction, etc. for smart and connected communities. Additional potential contributions of this study are summarized below:

i) Analysis of the reliability of the time-domain and frequency-domain HRV features extracted from the IHR signal.
ii) In-depth analysis of sleep HRV trends in healthy subjects providing novel insights.
iii) Preparation of a multi-modal scheme for on-device data processing and inference and analysis of its impact on system performance.
iv) Rank flu symptoms and elaborate on the concept of connected health using spatiotemporal visualization of EoI.
v) Prepare the resulting dataset, which contains novel aspects, in a way that it can be publicly available to interested parties to further research initiatives and advances.
vi) Overall, our intent is to better understand the prospects and challenges related to real-world implementation of sHealth and outline prospective solutions.

4.2 SYSTEM DESIGN

4.2.1 SCC Health Framework

We aimed to develop an end to end solution for sHealth. The system architecture of our developed SCC Health framework is shown in Figure 4.1 below. The main components in the framework are body-worn flexible IJP sensors (passive and battery-less), a scanner (reader

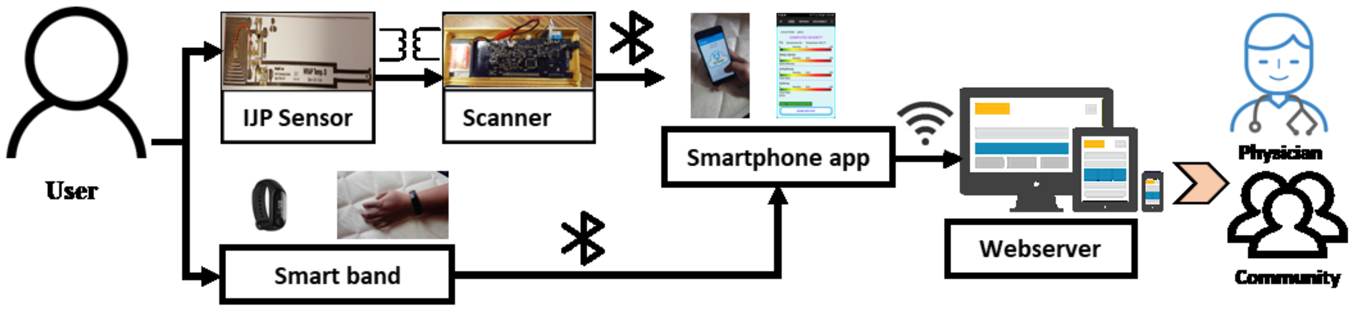


Figure 4.1: System architecture for the proposed SCC-Health framework.

for the passive sensors) on a printed circuit board, commercial wearables, a custom smartphone app (SCC-Health app), and a custom web server (SSC-Health server). For physiological data collection, we utilized both IJP sensors and commercial wearables, such as a smart wristband (Mi Band 2, Xiaomi) and a fingertip pulse oximeter (CMS50E, Crucial Medical Systems), as applicable. The IJP sensors are zero-power, analog, wireless, and fully passive. Data collected in the IJP sensors are pre-processed and digitized by the custom-made scanner [21]. Data from the scanner is transmitted to the smartphone via Bluetooth. Data reliability checks, feature extraction, and classification/regression using a pre-trained machine learning model are performed in the smartphone for disease detection and severity assessment. Computed severity of the disease is then visualized in the smartphone as well as shared with the webserver using a Wi-Fi/ cellular network for observing temporal and spatial distribution of the diseases.

4.2.2 Inkjet-Printed Sensors

The WRAP temperature sensor layout, shown in Figure 4.2(a), has been generated in Scalable Vector Graphics (SVG) file format using Inkscape, a free and open-source vector graphics editor. The layouts were exported to a PNG format with a resolution of 1693 dpi. The length of the designed sensors is 13 cm; although the length can be customized per need by pushing or drawing the NTC transducer away or closer. The IJP silver traces have a thickness in the range of 1-2 μm. It is possible to manipulate the trace thickness by changing the drop length and drop spacing of the inkjet printing. Reduced drop size results in thinner strains and vice versa. The entire circuit requires 1 discrete component of 3-pins, 7 discrete components of 2-pins, and 1 leaping wire. The components are a Qs-NPN Transistor (MMBTH10), D1, D2–Diodes (CDBF0130L), NTC–Negative Temperature Coefficient Transducer (NCP21XV103J03RA,10 kΩ), C1-, C2-Capacitors (0.47 μF), a C_B-Capacitor (0.1 μF), and a Cs-Capacitor (0.22 μF). The capacitor-diode pairs develop a 2-stage voltage doubler. When voltage is applied across the RC circuit formed by NTC and C_B, the NPN transistor Qs is turned on after a time delay and loads the inductive coil Ls with load impedance Cs and the emitter-collector resistance Qs. This loading effect can be detected at the scanner as a signal transition and the corresponding temperature can be calibrated based on the signal characteristics [2].

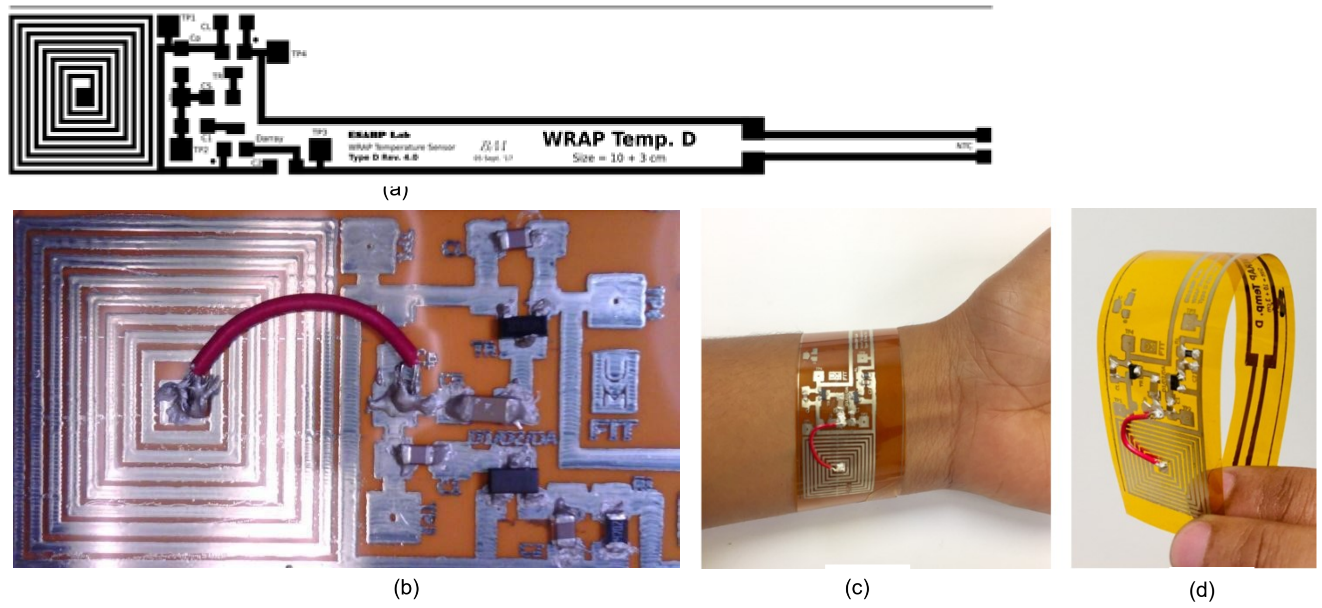


Figure 4.2: IJP WRAP Sensor (a) design, (b) microscopic view of printed sensor, (c) IJP sensor placed on the arm, and (d) flexibility of the sensor.

A Dimatix Material Deposition Printer (DMP-2831, Fujifilm, Dimatix Inc., NH) was used for manufacturing the designed sensor. MEMS-based nozzles enable the printer to achieve a high-resolution track fabrication (up to 20 µm) and precise control over deposition height. The printer ink was prepared using a combination of 25% Silver Ink (Ag-B25) (Metalon JS-B25HV, Novacentrix, Austin, TX), and Polypyrrole (PPy) (Sigma Aldrich, St Louis, MO) in a ratio of 1:1 and mixed well with a vortex mixer. The first step of the manufacturing technique was to print the silver traces on the polyimide (PI) tape (1 mil, Master Tape) substrate. This printing was performed with 10pL cartridges from Dimatix using all 16 nozzles., with resulting Ag/PPy traces on PI substrate being thermally cured at 250º C [22]. Discrete surface mount devices were then electronically attached to the printed traces using low-temperature curable silver epoxy (8331S, MG Chemicals, Surrey, BC, Canada) with 1:1 of Part A and Part B. A microscopic view of the manufactured sensor is shown in Figure 4.2 (b). Figures 4.2(c) and 4.2(d) show the high flexibility of the sensor.

4.2.3 Smartphone Application

The smartphone application SCC Health was developed using an android studio integrated development environment (IDE). The build tool version of android was 25.0.2 and minSdkVersion was 19. As shown in Figure 4.3, the main modules of the app are the signal processing module, questionnaire module, AI module, visualization module, a storage module, and networking module. The networking module enables the user to pair and connect with Bluetooth/ Bluetooth Low Energy(BLE) devices. It also has a Wi-Fi module to transfer data to the Web Server by means HyperText Transfer Protocol (HTTP) POST method. The signal processing module performs byte catenation, implements a moving average filter, extracts features, and performs data quality checks. The AI module loads the pre-trained predictive models from the asset directory and generates a prediction result based on the extracted features. It also has a mathematical model to map the EoI to a scale of 0-1. The visualization module uses GraphView (free library for android) to visualize the raw signal, temporal trends in EoI, etc. It also visualizes the computed EoI on a gradient scale. The electronic questionnaire module is used to collect user reported disease-related symptoms.

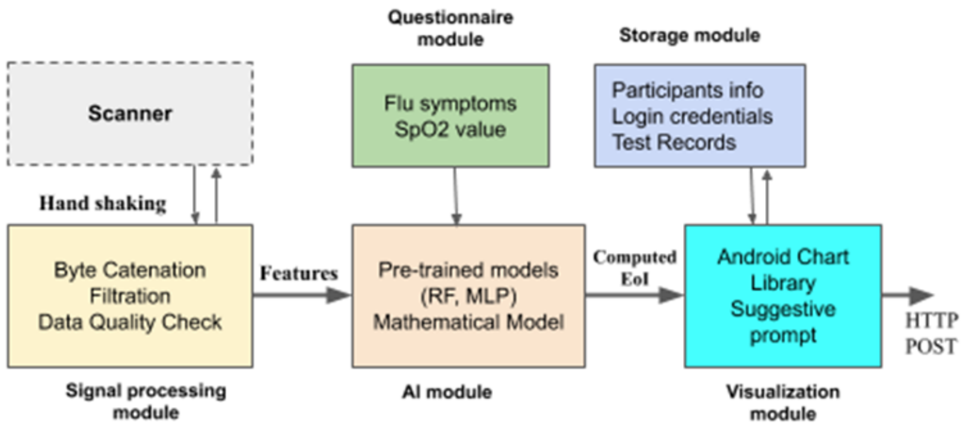


Figure 4.3: Component diagram of SCC Health application.

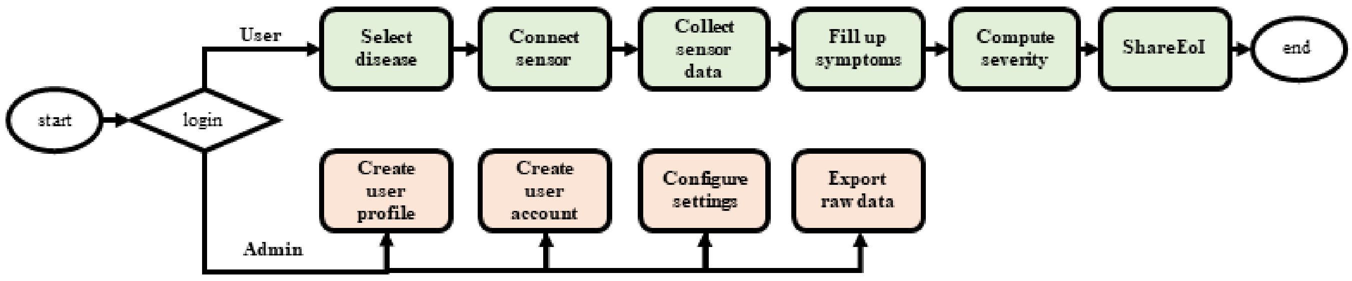


Figure 4.4: SCC Health application UX flow diagram.

The flow diagram for the SCC Health application is shown in Figure 4.4. The admin has a default username and password to login, but the user needs to acquire a personalized username and password provided by the admin. Before assigning a username and password, the admin records user information, including an ID and address by creating a profile for the user. The same app can be used to create profiles for multiple users and all user profiles are saved in an SQLite database. This is particularly helpful if all the members in a household do not own a personal device but would like to use the app. When user logs in with the correct username and password, the app greets him/her, and allows the user to proceed to the main menu where any of the three buttons may be chosen: About, Web Server, or Diagnostic. The About activity describes the details of the project. Selecting the Web Server activity allows the user to visit the SCC Health website. The Diagnostics activity enables access to the disease severity detection process, where the user may choose any one of the four diseases or select the "one-stop service" to test all of the diseases at once. Before selecting the aforementioned options, the user first needs to connect to the scanner via Bluetooth. For that, the user needs to click "Connect canner", which then provides a list of available Bluetooth devices. The user selects the scanner from this list. A status bar, located above the connect scanner button, indicates whether the app is connected to a scanner or not. After the connection is established, a list of five sensors appears, from which the user selects the one desired for collecting data. Upon clicking the "Collect data" button, the system starts collecting data from the sensor via the connected scanner. Prior to data collection, a handshaking protocol between the app and the scanner is executed where information about the type of disease, type of sensor, and duration of the scan is confirmed. Once the handshaking is successfully completed, the scanner powers up the WRAP sensor and starts collecting sensor data. The app displays a progress bar during data collection, and when data collection is completed for the sensor chosen, the app prompts the user to select the next sensor and collect data again. When the data have been collected from all of the sensors, the app prompts the user to select a severity-ranking algorithm, by clicking on "Compute severity". A data quality check is performed at this stage, mainly by confirming that the statistical features have values in the specified range. After performing the severity calculation, the app will show the sensor data values and the degree of severity. At this stage, the user has the option to save the test results and also to share the result with the SCC Health webserver. The app functionality has been tested thus far on various smartphones, including the Samsung Avant, Samsung Galaxy S6, and Samsung-SM-G90.

4.2.4 Web Server

Smartphone computed EoI's are dispatched to a cloud-hosted Web Server into JavaScript Object Notation (JSON) format. The front-end of the website was developed using HTML, CSS, and JavaScript. PHP was used for developing the backend. The HTTP POST request received at the server includes the participant's hash Id, region code, EoI, sickness type, date time, and algorithm type. The region codes are used instead of proper addresses of the users to ensure privacy. The smartphone app has the mechanism to convert the participant's domestic address into a region code during the time of entry. The region codes are mapped with reasonably large geographical locations. MySQL has been used for developing the database for the SCC Health server. This database has two tables: participants' facts and EoI statistics for four diseases—Arrhythmia, Chronic Obstructive Pulmonary Disease, Flu, and Sleep Apnea. In the backend, JSON data is encoded with PHP and inserted into the participants' table of the database. If two participants' ship statistics at the identical time, the server obeys the rule of FIFO (First-In-First-Out) and inserts the data that has arrived first, while retaining the remaining data in a queue. To enforce access control to SCC Health records three types of login credentials are utilized: 1) Admin: Admin login has full access to all data and different types of facts management. Admin has the additional capability to create login credentials for the participants. Admin can also add a downloadable Dalvik Executable (dex) file to be used for disorder severity estimation. 2) User: User login has the privilege to visualize the participants' statistics utilizing one spatial plot and two temporal plots. Login credentials are created by the admin for the recruited subjects only. 3) Guests: Guests have privileges equal to that of a user, with one exception. Guests can only observe the spatial and temporal plots with mock data. These mock facts are arbitrarily entered to test the functionality of the web visualization and are not collected from participants. No login credentials are required for guest login, so that anyone can experience the spatiotemporal visualization through this login [23].

4.3 METHODOLOGY

4.3.1 Data Collection

The Institutional Review Board (IRB) at the University of Memphis granted IRB approval for this study (IRB# PRO-FY2017-474). In collaboration with community partners, including a district of the United Methodist Church in Memphis, TN, USA, 9 participants were recruited for the study. Recruited participants were of different age and gender groups, with everyone being over 18 years old. Participants were encouraged to attend 2 sessions of data collection, where each session was 1-month long. The participants used a supplied senor suite for data collection during the 1-month session. The sensor suite details are as below:

i) android smartphone - 1

ii) IJP temperature sensor - 1

iii) Smart wristband - 1

iv) Finger pulse oximeter - 1

v) Scanner for IJP sensor - 1

Each participant attended a mandatory training session at the beginning of the session where the usage policy of the sensor suite was described and the entire testing procedure was demonstrated. Consultation was provided by a public health professional associated with the project when necessary. The protocol for data collection was vetted with physicians and clinical experts, with the measures collected listed below:

Step 1: Core body temperature (from armpit) using the IJP temperature sensor.

Step 2: SpO_2 using a finger pulse oximetry device immediately before sleep.

Step 3: Instantaneous heart rate using the wristband and the custom smartphone app.

Step 4: Continued HR data collection during the entire sleep period.

Step 5: SpO_2 assessed again using the finger pulse oximeter after waking up.

Step 6: Disease severity algorithms computed.

Step 7: Share the computed EoI results with the webserver for community-wide spatiotemporal visualization.

At the end of each session, a survey was administered and the raw sensor data collected in the smartphone was exported to a PC with consent from the participant for offline analysis. This was repeated for a total of 16 sessions. The study spanned from 06/11/2018 to 12/16/2019. A summary of the collected dataset is summarized in Table 4.1.

Table 4.1.: Summary of SCC-Health dataset.

	Count (N)	Standard Duration	Description
Participants	9	-	Age: 20 – 70, Gender: Male and Female, Race: Caucasian, Asian
Sessions	16	1 month	Home environment
IHR records	286	8 hours	Data were collected during sleep hours
Temperature records	375	20 ms	Data was collected before sleep
User Interaction Logs	16	1 month	User interaction with the app activities
EoI values	1008	-	Computed using pre-trained machine learning models

In total, 286 IHR records, 375 temperature (TP) data, and 1008 EoI scores recorded on a continuous scale of 0 to 1 rounded off to 2 decimal places (where 0 = low severity and 1 = high severity) for each disease of interest. Also, 16 event log files provided important insights regarding user interactions with the system. All data records were de-identified providing an uncorrelated and randomized participant ID. The geographical location of each participant was hash mapped with no option for back-tracing. The collected anonymized dataset will be made publicly available upon IRB approval.

4.3.2 Extraction and analysis of biomarkers

The introduction of pathological conditions in the human body is often reflected by a change in the vital signs [24-25]. Among the vital signs, heart rate, respiration rate, blood pressure, oxygen saturation level, body temperature, etc. have been used as important biomarkers for diseases like asthma, chronic obstructive pulmonary disease, obstructive sleep apnea, flu, etc. [26]. Change in vital signs can be attributed to an underlying change in the functionality of the autonomic nervous system. The autonomic nervous system (ANS) is part of the peripheral nervous system and plays an important role in balancing the internal environment of the body which includes heart rate, blood pressure, body temperature, coughing and sneezing, sexual arousal, oxygen, and Co2 level in the blood, etc. [27]. The regulations enforced by ANS take place involuntarily and without conscious effort. The autonomic nervous system has two main divisions- sympathetic and parasympathetic. Many organs are primarily controlled by either the sympathetic or parasympathetic division. The sympathetic division prepares the body for stressful or emergency, whereas the parasympathetic division prepares the body for rest and digest [28]. These two opposing divisions work together to establish a balance known as the autonomic balance. Autonomic balance plays a key role in sound health and wellness [29].

Heart rate variability (HRV) is an accurate non-invasive measure of the ANS function [30]. It is defined by a set of measures that describe the beat to beat variability of heartbeats. Traditionally, HRV is extracted from an ECG signal after due pre-processing. As of now, capturing of ECG using everyday wearables i.e. smart watch or smart band require manual intervention and are not autonomous [31]. Blood volume pulse (BVP) is widely used as a method of measuring the heart rate. The BVP measures heart rate based on the volume of blood that passes through the tissues in a localized area with each beat (pulse) of the heart. BVP measurement is obtained by the use of a photoplethysmography (PPG) sensor. This component measures changes in blood volume in the arteries and capillaries that correspond to changes in the heart rate and blood flow. The PPG sensor detects changes by shining an infrared light, typically via a light-emitting diode (LED), onto the surface of the body. This light is transmitted through the tissues, then backscattered and reflected by the tissue before reaching the photodetector of the PPG sensor. Red light is selectively absorbed by the hemoglobin of the red blood cells and reflected by other tissues. The amount of light that returns to the PPG photodetector is proportional to the relative volume of blood present in the tissue. The BVP amplitude is derived from the raw BVP signal and indicates relative blood flow. The heart rate (HR) is derived from the raw BVP signal by measuring the inter-beat interval i.e. distance between the peaks of the waveform. The estimation of HR from BVP is well investigated [32]. HR is not as informative as HRV and often fails to provide discriminatory information for disease classification or severity estimation when used alone [33]. The extraction of reliable HRV measures from PPG is challenging due to the high susceptibility of PPG to noise and motion artifacts. In this study, we extracted the time domain and frequency domain features of HRV from instantaneous heart rate (IHR) data collected via PPG during sleep. The method of HRV feature extraction has been shown in Figure 4.5.

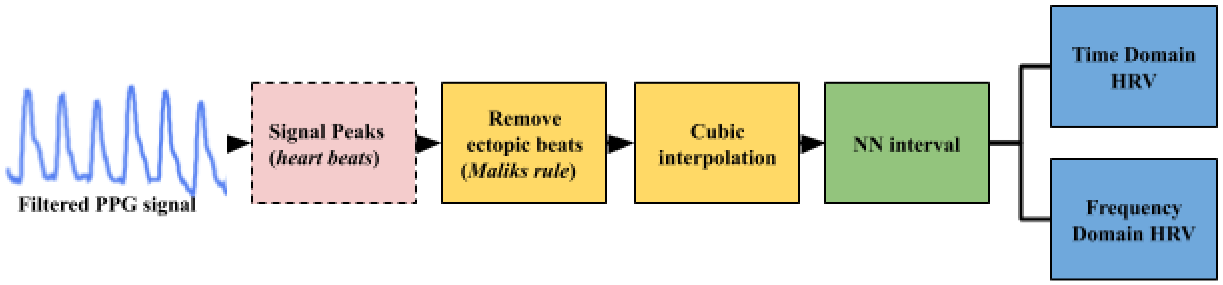


Figure 4.5: Method for HRV feature extraction from smart band data.

For HRV feature extraction we followed the standard HRV guideline [34]. The filtering of the raw PPG signal, detection of peaks, and computation of IHR are done by the algorithm embedded in the smart band by the manufacturer. Peak to peak interval has been computed from the IHR signal. Then Malik's rule has been used for removing the ectopic beats and followed by a cubic interpolation to interpolate the missing beats. The normal to normal (NN) interval obtained after the pre-processing has been used for extracting the time domain and frequency domain features of HRV. The estimation of HRV has been conducted in the smart phone. The extracted features have been defined in Table 4.2. For assessing the reliability of the extracted

Table 4.2. Time domain and frequency domain features of HRV from Smart band data

HRV Feature	Unit	Description
AVNN	ms	Mean of NN-interval
SDNN	ms	Standard deviation of all NN intervals.
SDANN	-	Standard deviation of the averages of NN intervals in all 5 min segments of the entire recording.
RMSSD	-	Defined as the square root of the mean of the squares of differences between adjacent NN intervals.
SDNN index	-	Mean of the standard deviation of all NN intervals for all 5 min segments of the entire recording.
VLF	s^2	Power in very low frequency range
LF	s^2	Power in low frequency range
HF	s^2	Power in low frequency range
Total Power	s^2	Variance of all NN intervals

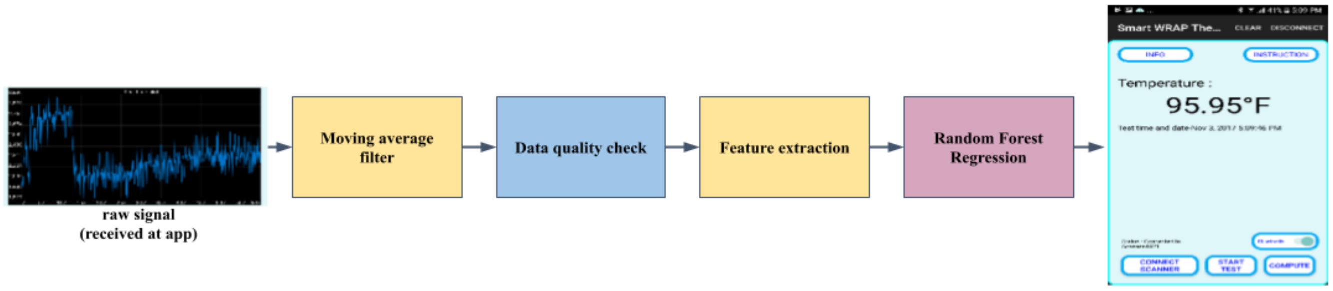


Fig. 4.6 Method of signal processing and feature extraction for IJP sensor characterization

measures we compared them with the normative range of HRV values when extracted from regular ECG as reported by other studies.

Core-body temperature is an important biomarker for the flu and some other diseases. To facilitate in-vitro temperature estimation, a controlled heat pad was used to generate temperatures in the range 69ºF - 107ºF. Initial labels for temperatures were obtained using a fiber optic thermometer and the IJP sensor response corresponding to each temperature was obtained. Five features are extracted from the signal, which is 1) amplitude average 2) skewness, 3) kurtosis, 3) range and 4) time delay of high to low transition. Then a Random Forest regression method has been trained and validated for temperature estimation using these features [35]. A similar process has been followed for in-vivo temperature estimation during the field study. The block diagram for body temperature monitoring using the app has been shown in Fig. 4.6. The raw signal received at the app from the temperature sensor contains high-frequency noise and oscillations. A moving average filter has been used to filter the signal, after that, a data check is performed. The metrics used for data checks are the mean and standard deviation of the range defined by the difference of maximum and minimum amplitude. Since we expect a transition from high to low as per sensor characteristics, if the data is valid, we get a good dip in the time series. For invalid data, either there is no transition or a low dip. Body temperature in addition to a set of user-reported symptoms have been used for ranking flu symptoms and estimating the symptoms severity of flu. The set of symptoms has been selected using a flu data set (UTMC, 386 records) available from the Influenza Research Database [36]. SelectKBest method from scikit-learn has been used for ranking the features and 5 top-ranked features have been used for the classification of flu [37].

4.3.3 Analysis of system performance for inference at the edge

We developed algorithms for the estimation of disease severity using HRV features, body temperature, and SpO_2 [38-39]. For algorithm development we considered separate datasets for chronic diseases (e.g. asthma, COPD, OSA) and acute conditions (Flu). WEKA has been used as the machine learning development environment. Trained and validated models have been exported from WEKA for integration in the android application. Details of the offline analysis for algorithm development and online inference scheme have been shown in Fig. 4.6. The pre-trained models embedded in the android application has been used for OSA and COPD severity estimation using the sleep HRV features and measured SpO_2. Data pre-processing, feature extraction, and conversion of features into attribute related file format (ARFF) have been done before making inference using the pre-trained models. Similarly, an electronic questionnaire has been used in the app for getting the flu symptoms as a user reported outcome. The algorithm estimated the severity of flu on a scale of 0-1 based on the body temperature, where a higher body temperature corresponds to a higher severity. High precision (±1°F) thermometer (Optocon FOTEMP with TS2 optical fiber probe, Weidmann Technologies, Germany)) for validating the temperature measurements before deploying the sensors. Similarly, for HRV we used Miband-2 smart band and compared with medical grade Omron 3 series (Model BP7100) measurements.

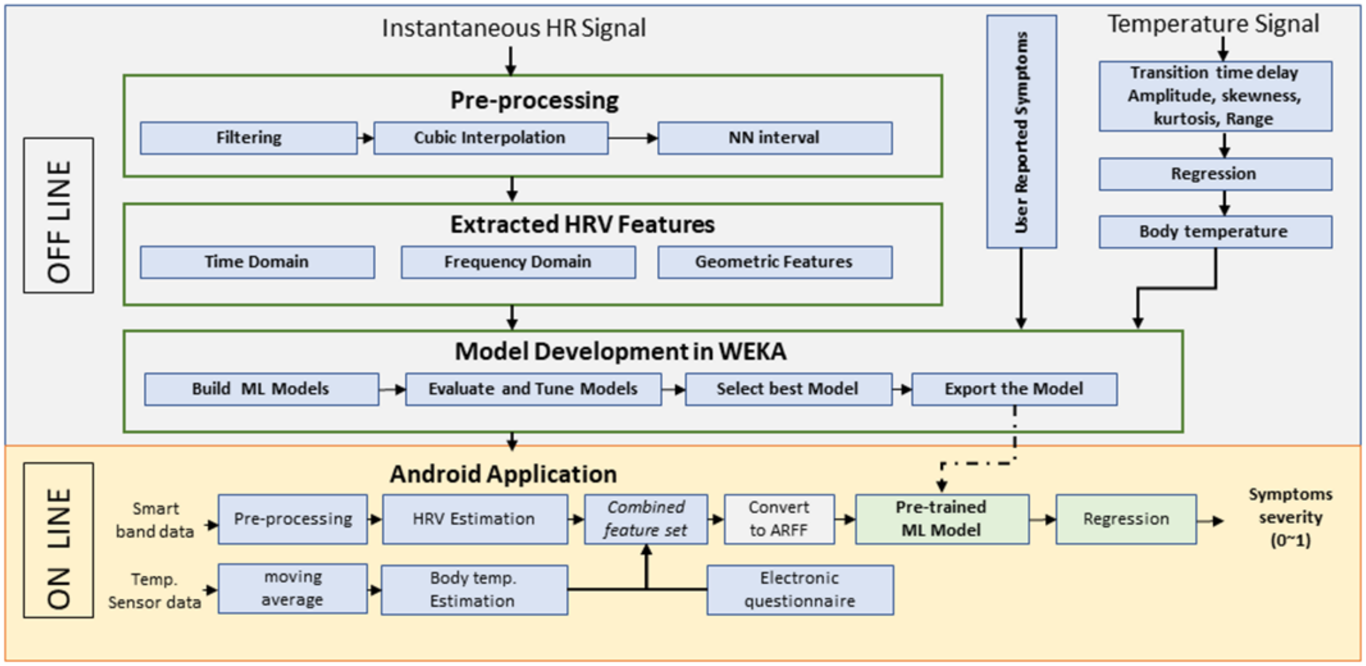


Fig. 4.7 Framework for pre-trained model development and integration in the app for making online inference at the edge

While the developed algorithms have been validated in the offline analysis, during the field study the aim was to evaluate the system performance for on-device data processing and machine learning. The main parameters that have been considered for evaluating the system performance are- power consumption, memory usage, storage capacity, and latency/ runtime. In addition to android reported values, we also used AcuBattery (Digibites, Netherlands) app for monitoring the performance parameters. We also investigated how on-device processing may help to reduce the overall cloud storage requirement by comparing the size of raw data to that of processed EoI values shared and stored in the Web Server.

4.3.4 Spatiotemporal visualization and human-technology interaction (HCI) analysis

The sharing of EoI for spatiotemporal visualization is fully voluntary and at the discretion of the user. For this study, the project area- Memphis statistical metropolitan area, has been divided into 19 grids with unique area code. Participants have been recruited from multiple grids. JavaScript object notation (JSON) has been used to share data from the android smartphone to the database in the webserver. The shared data contains participants' anonymized user ID, area code (hashed), computed EoI, the algorithm used for EoI computation, and date and time of data collection. For personalized monitoring of diseases, temporal trends of disease severity for a participant has been visualized using a time plot graph. Flow graph has been used for community health trend monitoring overtime where the Y-axis shows the no. of people affected with a disease and the X-axis shows the time intervals. In addition to that, a spatial plot has been used to visualize the severity of a disease in different areas at a period. Color coding has been used to indicate severity where red indicates the highest severity and green indicates the lowest severity. The EoI propagation-diagram in SCC-Health including spatiotemporal visualization pattern has been shown in Fig. 4.8. The equation used for the computation of the area severity is as below:

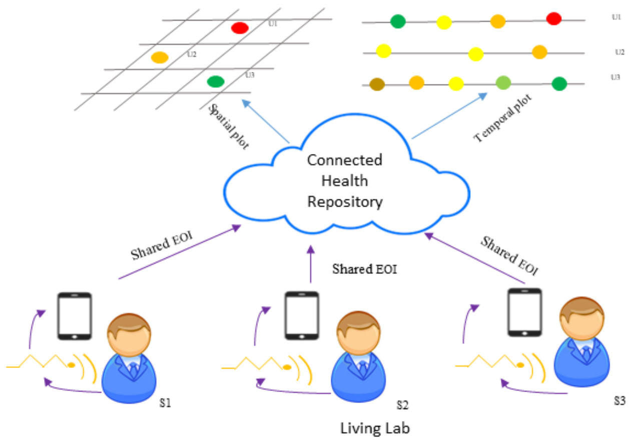


Fig. 4.8 EoI propagation in SCC-Health framework

Where k is the grid number, u is the participant’s number, N is the maximum number of participants in that kth grid who submit EoI for that particular disease and that particular time, and j is a participant who submits multiple measurements within that timeframe with the last submission as M. To investigate human-computer interaction some activities i.e. app login, sensor connection, data collection, and EoI sharing has been logged in the app as a CSV file with user consent. The logged information has been used to compute the drop-out ratio for subsequent action. Also, user interaction with the website has been logged in the webserver.

4.4 RESULTS AND DISCUSSION

Snapshots from the functional app have been shown in Fig. 4.9. The first screen from left shows the options the user may select on entering the app, the 2nd screen shows the test guidelines and brief instruction and information regarding the diseases of interest, the 3rd screen shows the option for selection of sensor that will be used for the physiological data collection, the 4th screen shows Progress Bar for monitoring runtime latency, the 5th screen shows the electronic questionnaire for collection user reported symptoms, and the 6th screen shows the visualization of the computed severity in a gradient scale.

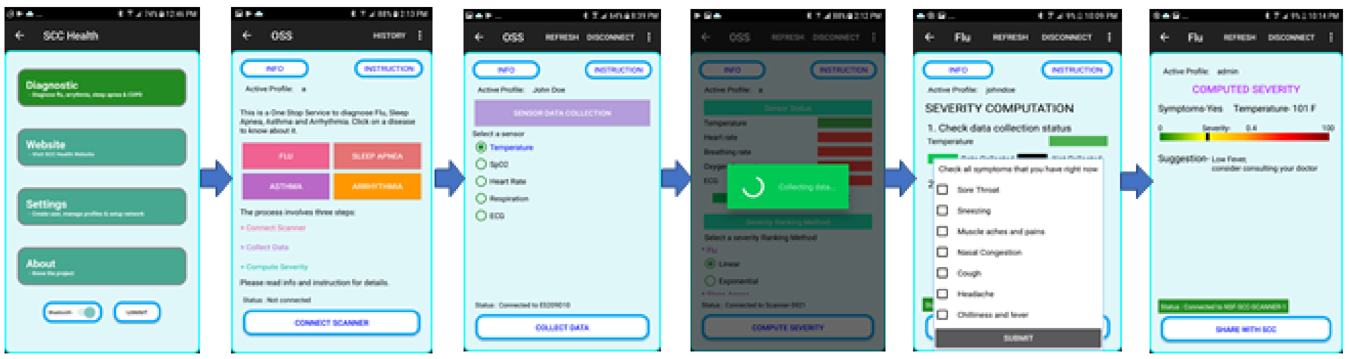


Fig. 4.9 Snapshots from the functional android app

Table 4.3 shows the time domain and frequency domain HRV feature values computed using the smartwatch data. It also shows a comparison of extracted HRV values with the normative values (clinically accepted) for healthy subjects [40-41]. The computed feature values are within or very close to the normal range which indicates the usability of the extracted HRV features as a reliable biomarker for the diseases of interest and general autonomic assessment. The trend line of the resting heart rate during sleep showed a hammock pattern for the healthy subjects as shown in Fig. 4.10. The HR gradually decreases as the sleep time increases and the sleep stage deepens. Before waking up the trend reverses and an upward trend becomes visible. The Poincare plot of NN intervals for the healthy subjects shows a comet pattern as shown in Fig. 4.10. An irregular pattern such as torpedo shape; fan shape or complex pattern is indicative of disease [42].

Table 4.3. Comparison of computed time domain and frequency domain HRV features with normative values

HRV Feature	HR (bpm)	AVNN (ms)	SDNN (ms)	pNN50 (%)	RMSSD (ms)	LFnu	HFnu	LF/HF
SCC Health Study	71.46 ± 8.72	866.77 ± 108.77	101.53 ± 39.63	24.25 ± 13.83	49.96 ± 17.9	71.73 ± 12.76	28.26 ± 12.76	5.80 ± 2.21
Normative Range	54 - 102	785 - 1160	79 - 219	1 - 48	15 - 63	30-65	16-60	1.1 – 11.6

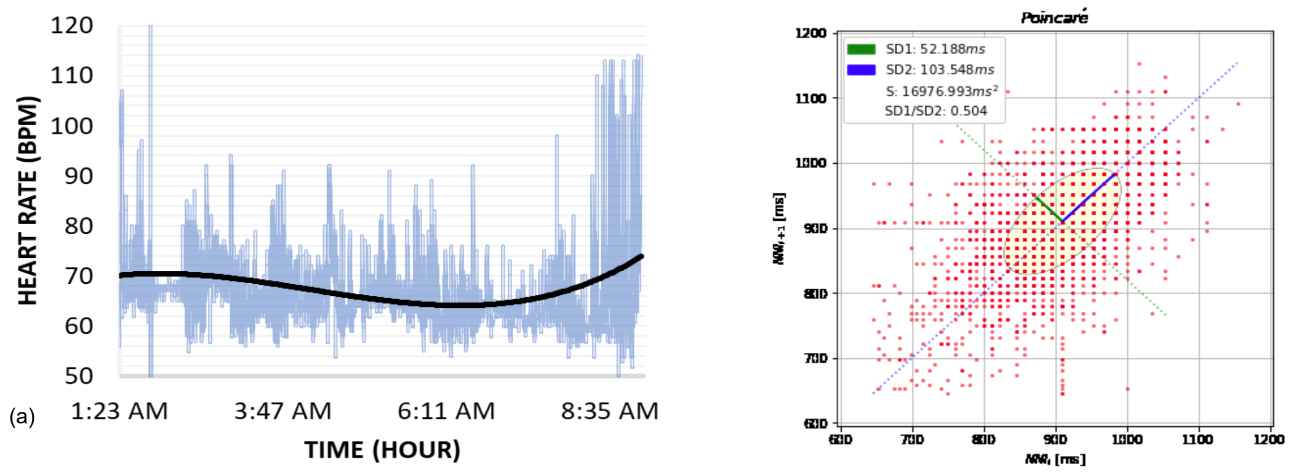


Fig. 4.10 Sleep HRV (a) trend line and (b) Poincare plot for healthy subjects

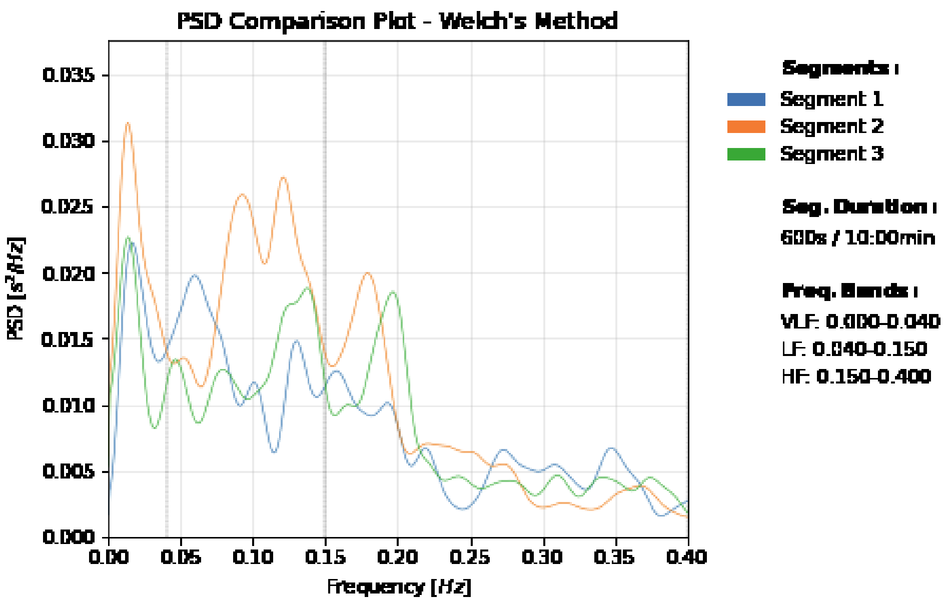


Fig. 4.11 Power Spectral density at different episodes of sleep hours

The power spectral density at the beginning, middle, and at the end of the entire sleep duration has been visualized in Fig. 4.11. A similar pattern is observed in all the three intervals i.e. there is a high power density at very low frequency (VLF) and low power density at the high-frequency band. However, power spectral density at low frequency (LF) band becomes more prominent during the middle of the sleep. The empirical cumulative distribution function (ECDF) shows a higher variability in the resting HR of young people compared to the old one (Fig. 4.12(a)). The t-SNE visualization of the HRV score shows a clustering pattern for age as shown in Fig. 4.12(b).

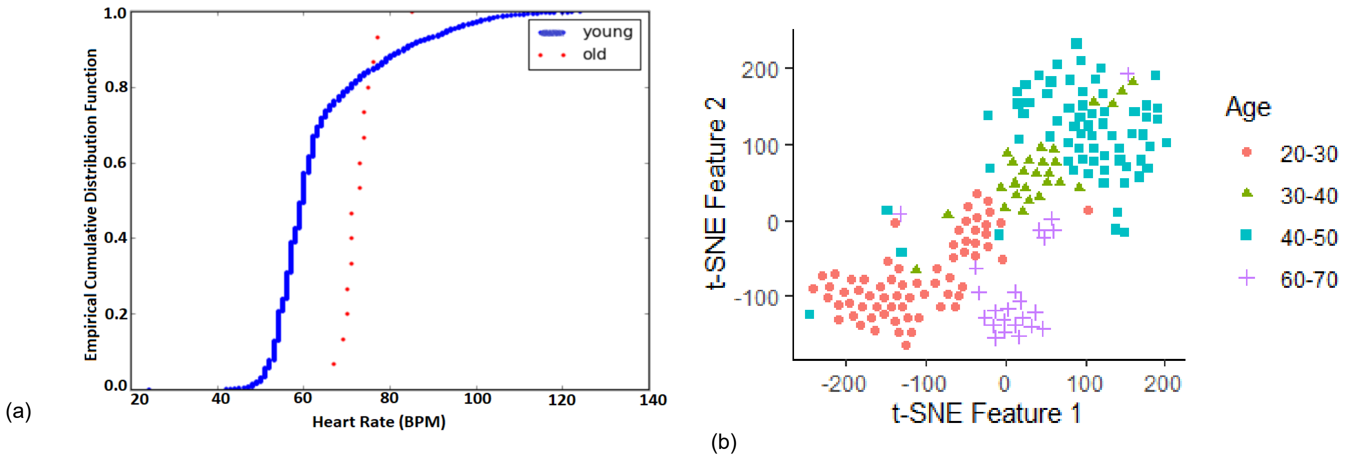


Fig. 4.12 a) ECDF of resting heart rate for age categories b) clustering pattern of HRV scores for age categories

Table 4.4. Comparison of female and male HRV

HRV Measure	Female (Median)	Male (Median)
Mean HR (bpm)	75.33	67.99
SDNN (ms)	78.63	101.87
RMSSD (ms)	43.60	60.17
pNN50 (%)	19.03	33.19
pNN20 (%)	48.81	61.27
LF/HF	3.16	2.55
SD1	30.84	42.57
SD2	106.102	138.33
SD1/SD2	3.86	3.57
Triangular Index	7.60	8.90
Sample Entropy	0.75	0.93

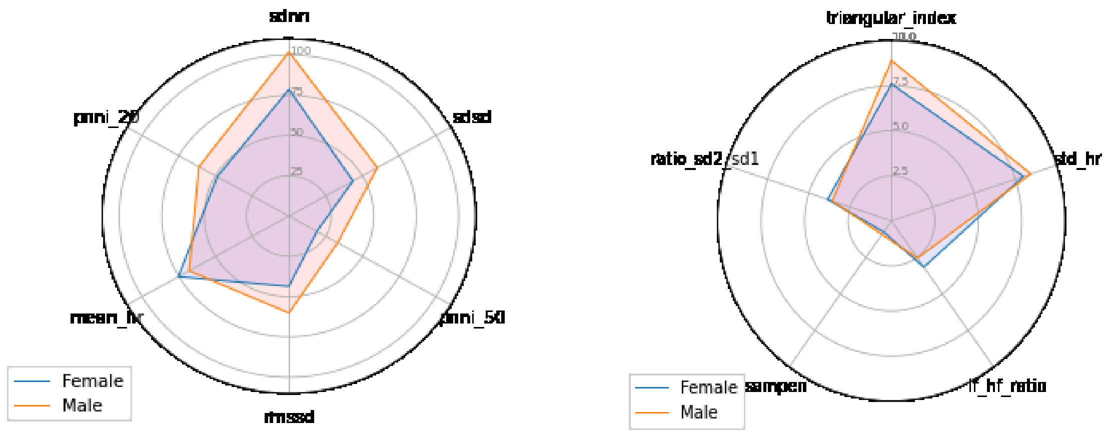


Fig. 4.13 (a) (b) Radar chart comparison of prominent HRV measures of the female with male The median sleep HR of women has a higher value than the median sleep HR of men.

Table. 4.4 shows the median value of time-domain, frequency domain, and geometric HRV measures for the male and female categories. A radar chart comparison has been made to visualize the difference between the prominent sleep HRV measures of females with that of males. As shown in Fig. 4.13 (a), sdnn, sdsd, rmssd, pnn20, and pnn50 have a higher value for the male group than the female group. As shown in Fig. 4.13(b), triangular index of HRV have a higher value in male but the ratio of LF power to HF power has higher values for the females than males. The Empirical Cumulative Distribution Function (ECDF) for median HR during the weekdays and weekend has been shown in Fig. 4.14. While resting median HR during sleep for the weekend is slower than weekdays, the difference is not significant (p-value>.05) in t-test.

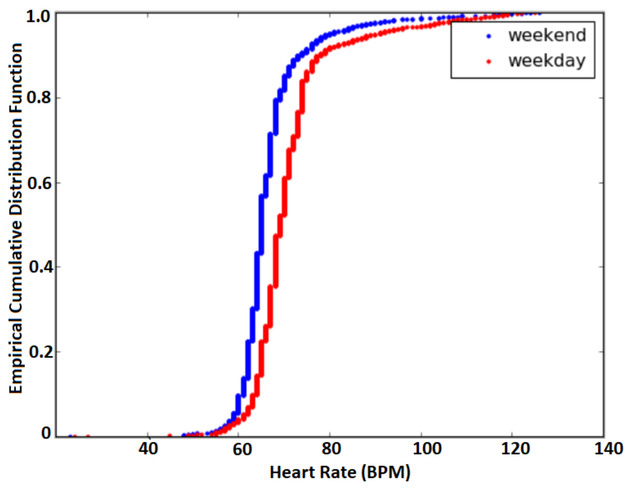


Fig. 4.14 Comparison of sleep HR during weekdays and weekends

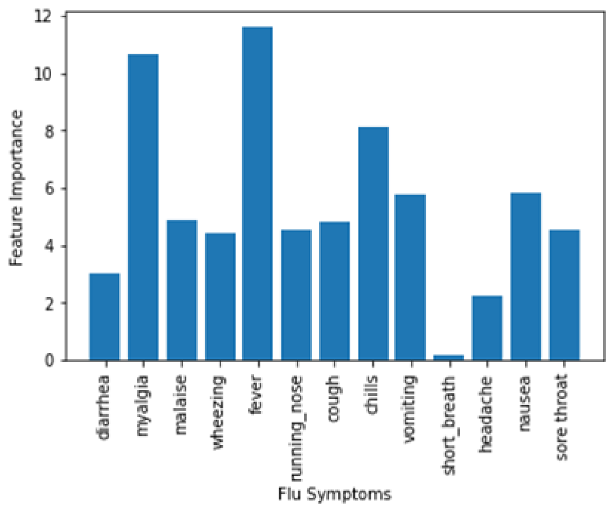
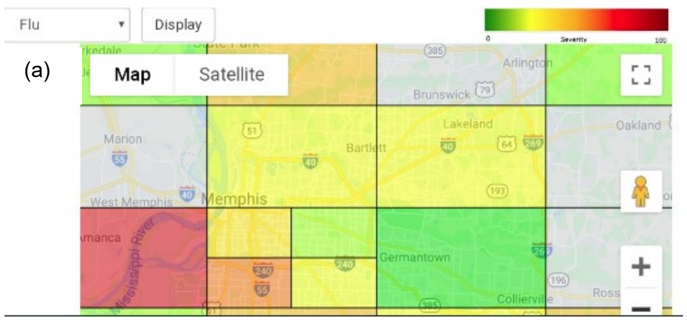


Fig. 4.15 Ranking of flu symptoms in terms of feature importance

The feature importance of flu symptoms has been shown in Fig. 4.15. The most 5 important features are - fever, myalgia, chills, nausea, vomiting, and sore throat. The spatial and temporal plots of symptom severity have been shown in Fig. 4.16. The gradient scale indicates the severity from a scale of 0-1. Green corresponds to a low severity value whereas red corresponds to a high severity value. The spatial plot is an aggregate of the severity of all the subjects within that spatial grid on a specific date, and the temporal plot is indicative of the severity of the disease-related symptoms of a person over time. The plots can be animated to track the progression of the disease over time in a specific region of interest.



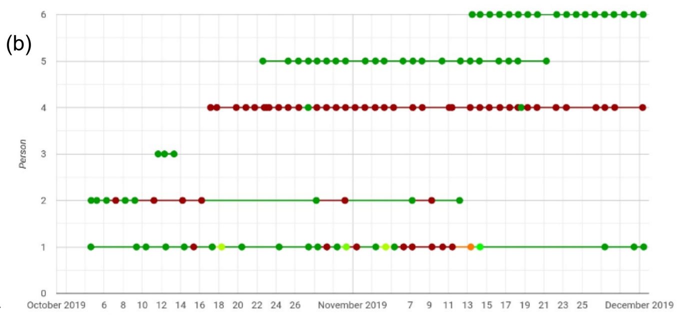


Fig. 4.16 Snapshots showing (a) Spatial distribution of symptoms severity (b) Temporal trends of symptoms severity over time from data collected from real-life "livings labs" of participants in this study.

The results for evaluating system performance have been shown in Table 4.5. Samsung Galaxy J3 Orbit (Samsung Corp.) has been used as the test environment. While the power consumption of a smartphone is highly impacted by other factors including screen, in our study data has been collected for the entire night (for IHR signal) without running the smartphone out of charge. If the power consumption of only this app is considered, it is estimated that the battery will support sound for 50 hours of continuous operation. The memory requirement has been reported for active data processing and computation including prediction using the pre-trained models. The memory requirement is only 9.4 MB which is approximately 2.5% of that of the device under test. The storage size is the sum of all the CSV files (TP sensor data-30, IHR signal-30) stored in the smartphone for an entire session of 30-days of data collection. The latency is small enough to support real-time processing and inference at the edge.

Table 4.5. Results of system performance evaluation

	Key Performance Indicator	Designed Capacity	SCC Health App
1	Battery	2600 mAH	51.9 mAH
2	Memory	2 GB	9.4 MB
3	Storage	16 GB	2.2 MB/session
4	Latency	-	500 ms

On-device data processing and inference lead to a reduction of the storage requirement in the cloud. The order of reduction achieved in our case is ~ 1000 as shown in Fig. 4.16(a). The data size in the edge device is the sum of all the raw data collected during all the sessions and for all the subjects, whereas the data size in the cloud is the size of the file containing EoI records obtained as a result of inference at the edge. This data reduction is critical in population-level data collection, where there will be a massive amount of data flow to the cloud thereby inducing a substantial amount of burden on the bandwidth requirement. A drop out in the user activities has been observed by analyzing the log data. As shown in Fig. 4.16(b) the dropout rate for the complete cycle completion is around 40%. User interaction with the website indicates a higher interaction during the time of active sessions compared to the rest of the study period.

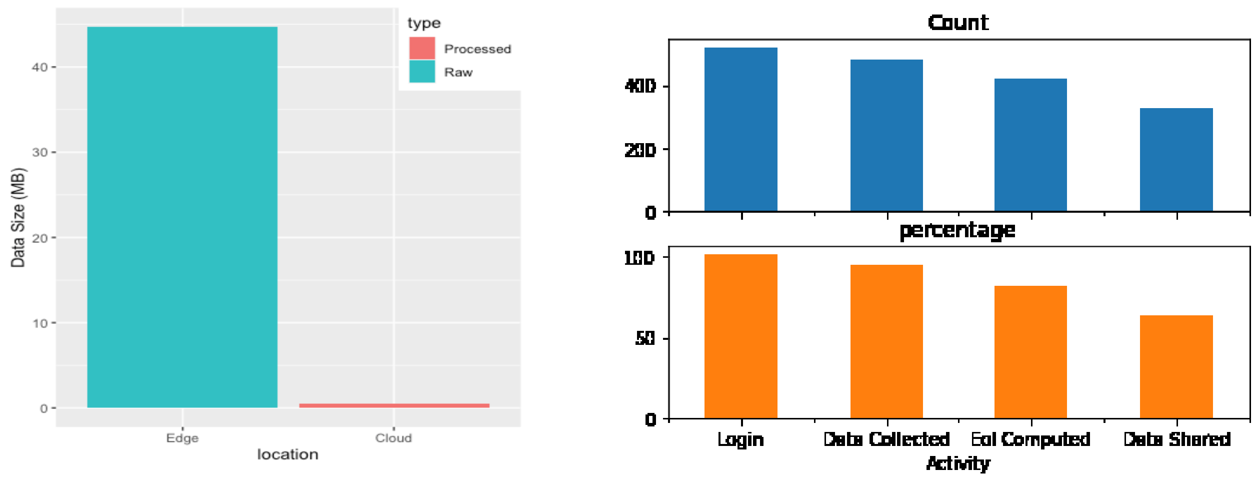


Fig. 4.16 a) plot for the order of data reduction b) statistics of human technology interaction

4. 5 LIMITATION AND FUTURE WORK

The challenging part of this study was the recruitment of subjects from the general population excluding the university community. One of the reasons is the commitment to use the device and collect data on a daily basis for a month-long session which is difficult for many people. The low number of subjects may also be an indication that human-technology interaction is still a taboo among the common people. We also observed that developing an iPhone app to recruit iPhone users as volunteers may help to increase the number of participants. While the study and the analysis revealed some novel insights and trends regarding the vital signs of the subjects it was not possible to identify or pinpoint the number of exacerbation events for asthma or COPD. A more meaningful and convincing result may be achieved from a similar study with a large number of subjects and doing a parallel medical check-up for the subjects on a regular basis and then correlating the observed trends with the observation of the medical practitioners. From a technology point of view, while polyamide based IJP sensor were and comfortable to the users, the range of low energy Bluetooth imposed some interruptions on continuous data transfer from the sensors to the smartphones. Specially, when the smartphone was left on the bedside and the participant went downstair or far-away corner of his house. Existence of low-quality data captured by the sensors reveal the importance of developing an online data reliability assessment algorithm to avoid misleading conclusions regarding the symptom's severity.

4.6 CONCLUSION

In this yearlong study, we investigated the real-world implementation challenges of a smart health solution using flexible, user-friendly wearables. We addressed the extraction of reliable biomarkers, on-device processing, and inference at the edge. We identified the challenges, human-technology interaction, and outlined prospective solutions. This paper outlines key technological outcomes from this pilot study. The results indicate the possibility of using everyday wearables and other sensors (e.g., IJP) for capturing events of interest (EoI), such as biomarkers for diseases as well as the prospect of on-device processing and machine learning at edge devices without impacting device performance adversely. We believe that SCC Health will help in the early identification of diseases and may also be useful in the spatiotemporal monitoring of pandemics, such as COVID-19.

REFERENCES

[1] Kumar, N., Akangire, G., Sullivan, B., Fairchild, K., & Sampath, V. (2019). Continuous vital sign analysis for predicting and preventing neonatal diseases in the twenty-first century: big data to the forefront. Pediatric research, 1-11, Springer Nature.

[2] Park Y. T. (2016). Emerging New Era of Mobile Health Technologies. Healthcare informatics research, 22(4), 253–254. https://doi.org/10.4258/hir.2016.22.4.253.

[3] Solanas, A., Patsakis, C., Conti, M., Vlachos, I. S., Ramos, V., Falcone, F., ... & Martinez-Balleste, A. (2014). Smart health: a context-aware health paradigm within smart cities. IEEE Communications Magazine, 52(8), 74-81.

[4] Leroy, G., Chen, H., & Rindflesch, T. C. (2014). Smart and Connected Health [Guest editors' introduction]. IEEE Intelligent Systems, 29(3), 2-5.

[5] Rani, S., Ahmed, S. H., & Shah, S. C. (2018). Smart health: a novel paradigm to control the chickungunya virus. IEEE Internet of Things Journal, 6(2), 1306-1311.

[6] Pathinarupothi, R. K., Durga, P., & Rangan, E. S. (2018). Iot-based smart edge for global health: Remote monitoring with severity detection and alerts transmission. IEEE Internet of Things Journal, 6(2), 2449-2462.

[7] Ismail, W. N., Hassan, M. M., Alsalamah, H. A., & Fortino, G. (2020). CNN-Based Health Model for Regular Health Factors Analysis in Internet-of-Medical Things Environment. IEEE Access, 8, 52541-52549.

[8] Singh, A., & Chatterjee, K. (2017). Cloud security issues and challenges: A survey. Journal of Network and Computer Applications, 79, 88-115.

[9] Verma, P., & Sood, S. K. (2018). Fog assisted-IoT enabled patient health monitoring in smart homes. IEEE Internet of Things Journal, 5(3), 1789-1796.

[10] Hu, J., Wu, K., & Liang, W. (2019). An IPv6-based framework for fog-assisted healthcare monitoring. Advances in Mechanical Engineering, 11(1), 1687814018819515.

[11] Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K., & Zhang, J. (2019). Edge intelligence: Paving the last mile of artificial intelligence with edge computing. Proceedings of the IEEE, 107(8), 1738-1762

[12] Chen, J., & Ran, X. (2019). Deep learning with edge computing: A review. Proceedings of the IEEE, 107(8), 1655-1674.

[13] Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. IEEE internet of things journal, 3(5), 637-646.
[14] Mudassar, B. A., Ko, J. H., & Mukhopadhyay, S. (2018). Edge-cloud collaborative processing for intelligent internet of things: A case study on smart surveillance. In 2018 55th
[15] Temko, A. (2017). Accurate heart rate monitoring during physical exercises using PPG. IEEE Transactions on Biomedical Engineering, 64(9), 2016-2024.
[16] Chiauzzi, E., Rodarte, C., & DasMahapatra, P. (2015). Patient-centered activity monitoring in the self-management of chronic health conditions. BMC medicine, 13(1), 1-6.
[17] El-Amrawy, F., & Nounou, M. I. (2015). Are currently available wearable devices for activity tracking and heart rate monitoring accurate, precise, and medically beneficial? Healthcare informatics research, 21(4), 315-320.
[18] Ernst, G. (2017). Heart-rate variability—More than heart beats?. Frontiers in public health, 5, 240.
[19] Li, J., Rossignol, F., & Macdonald, J. (2015). Inkjet printing for biosensor fabrication: combining chemistry and technology for advanced manufacturing. Lab on a Chip, 15(12), 2538-2558.
[20] Zaman, M. S., & Morshed, B. I. (2018). Design and Verification of a Portable Scanner for Body-Worn Wireless Resistive Analog Passive (WRAP) Sensors. In 2018 IEEE International Conference on Electro/Information Technology (EIT) (pp. 0548-0553). IEEE.
[21] Morshed, B. I., Harmon, B., Zaman, M. S., Rahman, M. J., Afroz, S., & Rahman, M. (2017). Inkjet printed fully-passive body-worn wireless sensors for smart and connected community (SCC). Journal of Low Power Electronics and Applications, 7(4), 26.
[22] Mohapatra, A., Morshed, B. I., Shamsir, S., & Islam, S. K. (2018, March). Inkjet printed thin film electronic traces on paper for low-cost body-worn electronic patch sensors. In 2018 IEEE 15th International Conference on Wearable and Implantable Body Sensor Networks (BSN) (pp. 169-172). IEEE.
[23] Afroz, S., & Morshed, B. I. (2018). Web Visualization of Temporal and Spatial Health Data from Smartphone App in Smart and Connected Community (SCC). In 2018 IEEE International Smart Cities Conference (ISC2) (pp. 1-6). IEEE.

[24] Brekke, I. J., Puntervoll, L. H., Pedersen, P. B., Kellett, J., & Brabrand, M. (2019). The value of vital sign trends in predicting and monitoring clinical deterioration: A systematic review. PloS one, 14(1), e0210875.

[25] University of Bristol. (2011). Using vital signs to predict severity of illness in children. ScienceDaily. Retrieved July 2, 2020 from www.sciencedaily.com/releases/2011/07/110706195858.htm

[26] Roche, F., Gaspoz, J. M., Court-Fortune, I., Minini, P., Pichot, V., Duverney, D., ... & Barthélémy, J. C. (1999). Screening of obstructive sleep apnea syndrome by heart rate variability analysis. Circulation, 100(13), 1411-1415.

[27] Donkelaar H.J. (2011) The Autonomic Nervous System. In: Clinical Neuroanatomy. Springer, Berlin, Heidelberg

[28] Appenzeller O (ed) (1999) The autonomic nervous system. Part I. Normal functions, vol 74, Handbook of clinical neurology. Elsevier Science, Amsterdam

[29] Appenzeller O, ed (2000) The autonomic nervous system. Part II. Dysfunctions. Handbook of clinical neurology, Vol 75. Elsevier Science, Amsterdam

[30] Sztajzel J. (2004). Heart rate variability: a noninvasive electrocardiographic method to measure the autonomic nervous system. Swiss medical weekly, 134(35-36), 514–522.

[31] Dyer, O. (2018). Sixty seconds on... Apple ECG.

[32] Jo, E., Lewis, K., Directo, D., Kim, M. J., & Dolezal, B. A. (2016). Validation of biofeedback wearables for photoplethysmographic heart rate tracking. Journal of sports science & medicine, 15(3), 540.

[33] Malik, M., Bigger, J. T., Camm, A. J., Kleiger, R. E., Malliani, A., Moss, A. J., & Schwartz, P. J. (1996). Heart rate variability: Standards of measurement, physiological interpretation, and clinical use. European heart journal, 17(3), 354-381.

[34] Rahman, M. J., & Morshed, B. I. (2019). Improving Accuracy of Inkjet Printed Core Body WRAP Temperature Sensor Using Random Forest Regression Implemented with an Android App. In 2019 United States National Committee of URSI National Radio Science Meeting (USNC-URSI NRSM) (pp. 1-2). IEEE.

[35] Squires, R. B., Noronha, J., Hunt, V., García-Sastre, A., Macken, C., Baumgarth, N., ... & Ramsey, A. (2012). Influenza research database: an integrated bioinformatics resource for influenza research and surveillance. Influenza and other respiratory viruses, 6(6), 404-416.

[36] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. the Journal of machine Learning research, 12, 2825-2830.
[37] Rahman, M. J., Mahajan, R., & Morshed, B. I. (2018, March). Severity classification of obstructive sleep apnea using only heart rate variability measures with an ensemble classifier. In 2018 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI) (pp. 33-36). IEEE.
[38] Siddiqui, T., & Morshed, B. I. (2018). Severity Classification of Chronic Obstructive Pulmonary Disease and Asthma with Heart Rate and SpO_2 Sensors. In 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC) (pp. 2929-2932). IEEE.
[39] Frank, E., Hall, M., Holmes, G., Kirkby, R., Pfahringer, B., Witten, I. H., & Trigg, L. (2009). Weka-a machine learning workbench for data mining. In Data mining and knowledge discovery handbook (pp. 1269-1277). Springer, Boston, MA.
[40] Umetani, K., Singer, D. H., McCraty, R., & Atkinson, M. (1998). Twenty-four-hour time domain heart rate variability and heart rate: relations to age and gender over nine decades. Journal of the American College of Cardiology, 31(3), 593-601.
[41] Shaffer, F., & Ginsberg, J. P. (2017). An overview of heart rate variability metrics and norms. Frontiers in public health, 5, 258.
[42] Khandoker, A. H., Karmakar, C., Brennan, M., Palaniswami, M., & Voss, A. (2013). Poincaré plot methods for heart rate variability analysis. Boston, MA, USA: Springer US. ACM/ESDA/IEEE Design Automation Conference (DAC) (pp. 1-6). IEEE.

Chapter 5
CONCLUSIONS AND FUTURE DIRECTIONS

5.1 KEY RESULTS:

The outcome of this research can be summarized as below:

1. We were able to develop a minimalist method using a deep artificial neural network for early estimation of obstructive sleep apnea severity as well as continuous monitoring in home environments. The off-line study results indicate that by using computationally inexpensive features from HRV and SpO_2, an area under the curve of 0.91 and an accuracy of 83.97% can be achieved for the severity classification of OSA. For estimation of the apnea-hypopnea index, an accuracy of RMSE=4.6 and R-squared value=0.71 was achieved in the test set using only ranked HRV and SpO_2 features. The method was integrated in a smartphone application and deployed with real-world subjects during a pilot study.

2. We addressed the need for developing a wearable sensor-based objective assessment method for estimation of sleep deficiency severity and developed a regression model for quantifying sleep deficiency severity using user-friendly wearables. The developed method achieved a performance of RMSE = 5.47 and R-squared value of 0.67 for sleep deficiency severity estimation and outperformed conventional methods; e.g., Functional Outcome of Sleep Questionnaire and Epworth Sleepiness Scale for assessing the impact of sleep apnea on sleep deficiency. Moreover, the results help pave the way for reliable and interpretable sleep deprivation severity estimation using a wearable device.

3. We developed an sHealth framework which included AI-enabled light-weight, low-resource algorithms for early detection of disease. The framework incorporated sensor-edge-cloud data flow, data privacy, and spatiotemporal visualization of events of interest in the cloud for community wide health monitoring. We also analyzed the human technology interaction in sHealth, identified the challenges for the real world implementation of sHealth solution, and outlined possible solutions. The framework has been tested with real-world subjects during a yearlong pilot study. When analyzed, the collected biomarkers supported our hypothesis that low-cost IJP sensors have potential for providing reliable data.

5.2 FUTURE RESEARCH DIRECTIONS

The proposed sHealth framework aims to deliver improved and ubiquitous healthcare for early detection of disease, pre-screening, and continuous monitoring of symptoms. One of the key focus areas of sHealth is community health, where users and stakeholders will have an opportunity to collaborate, formulate policy, and harness benefits from voluntarily shared information. The key challenges are extraction of reliable biomarkers from the data collected using IoT devices, seamless data transfer to a computing facility, making inference using artificial intelligence enabled methods, while maintaining user data privacy and system security. Incorporation of IoT-based low cost and user friendly wearable sensing technologies can help meet these challenges.

Although we developed an sHealth solution comprising of algorithms and framework and tested it in a pilot project, the framework nees to be deployed at a large scale in a greater geographical entity with more subjects. Also, parallel health evaluation of the participants during the study period by medical practitioners will help to evaluate the clinical value of the sHealth solution.

APPENDIX

A1. the source code for the smartphone app is available at https://github.com/esarplab/SCCHealth_v2.0. In addition, the IJP sensor design, disease severity estimation algorithms are also available as open source at https://github.com/esarplab/SCCHealth-MEMPHIS

Flowchart and sequence diagram for the SCC-Health app has been shown below-

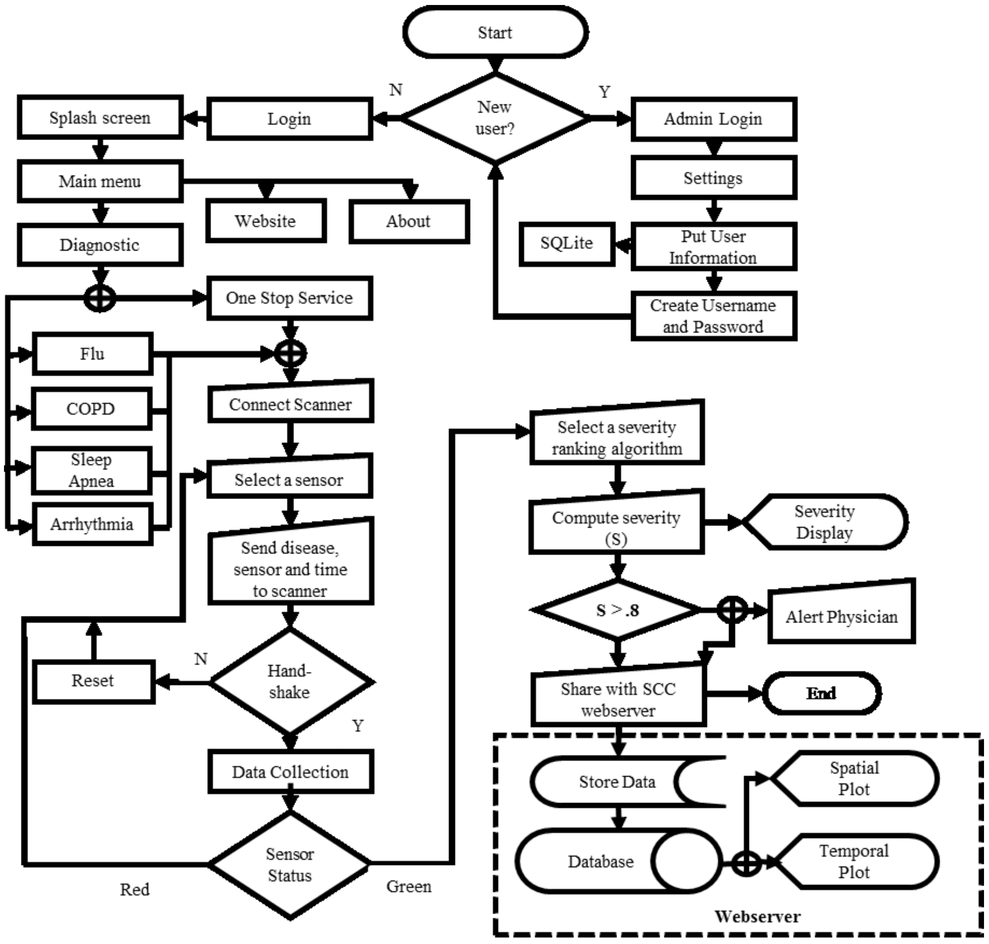


Fig. Flowchart for SCC Health android application

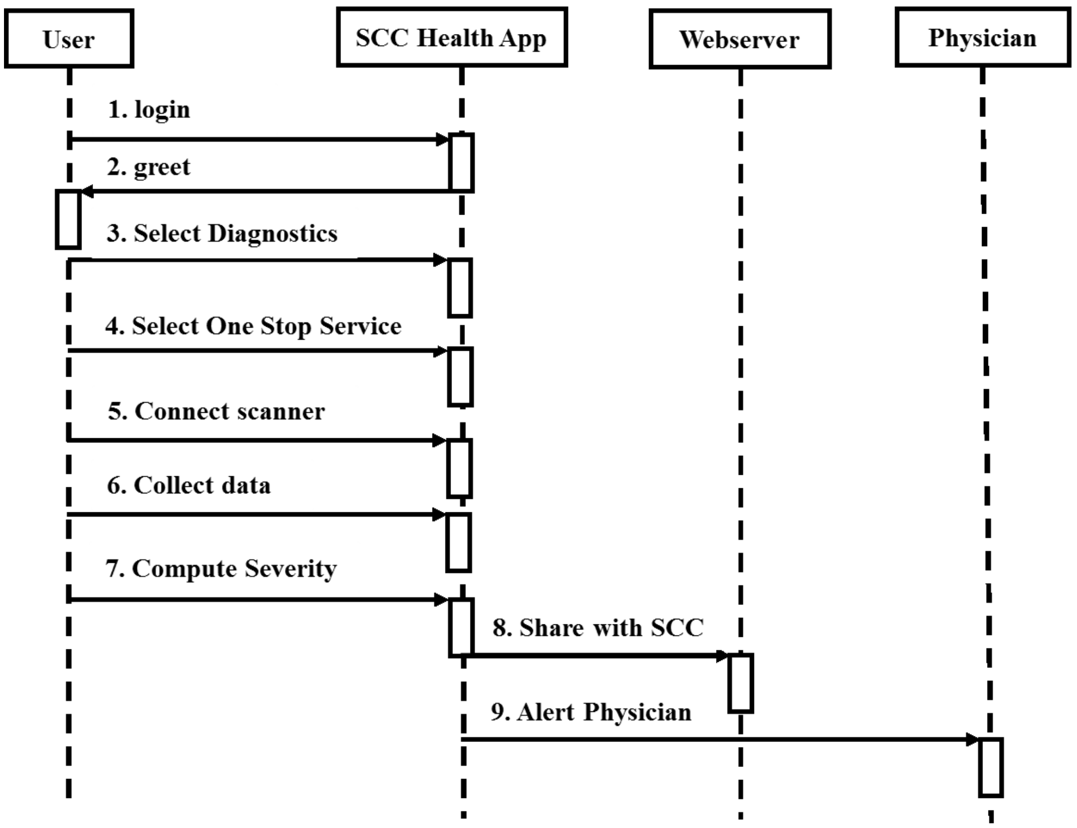
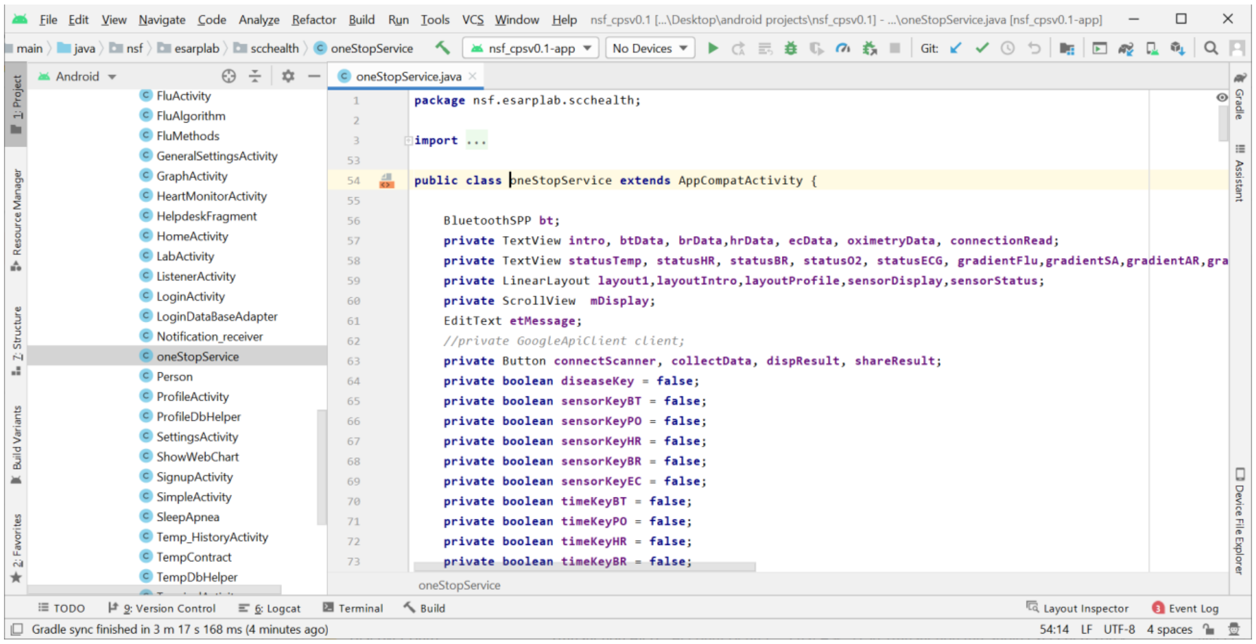


Fig. Sequence diagram for the SCC-Health app

Few Code snippets from the SCC-Health app has been given below:



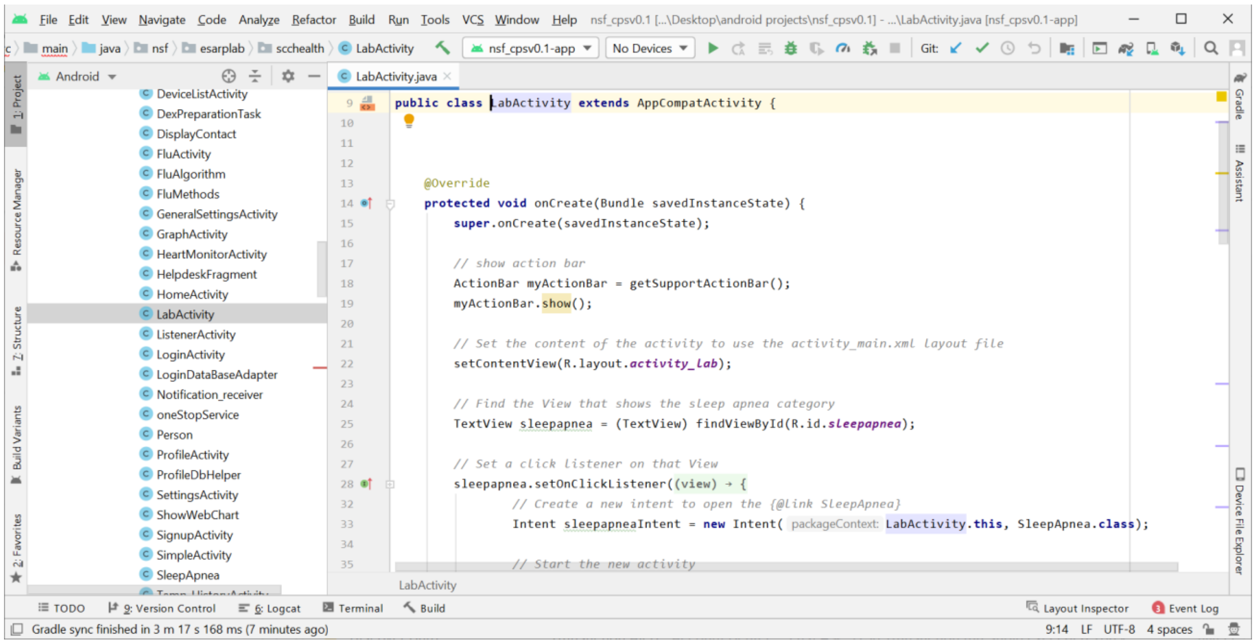
```java
package nsf.esarplab.scchealth;

import ...

public class oneStopService extends AppCompatActivity {

    BluetoothSPP bt;
    private TextView intro, btData, brData,hrData, ecData, oximetryData, connectionRead;
    private TextView statusTemp, statusHR, statusBR, statusO2, statusECG, gradientFlu,gradientSA,gradientAR,gra
    private LinearLayout layout1,layoutIntro,layoutProfile,sensorDisplay,sensorStatus;
    private ScrollView  mDisplay;
    EditText etMessage;
    //private GoogleApiClient client;
    private Button connectScanner, collectData, dispResult, shareResult;
    private boolean diseaseKey = false;
    private boolean sensorKeyBT = false;
    private boolean sensorKeyPO = false;
    private boolean sensorKeyHR = false;
    private boolean sensorKeyBR = false;
    private boolean sensorKeyEC = false;
    private boolean timeKeyBT = false;
    private boolean timeKeyPO = false;
    private boolean timeKeyHR = false;
    private boolean timeKeyBR = false;
```

Gradle sync finished in 3 m 17 s 168 ms (4 minutes ago)



```java
public class LabActivity extends AppCompatActivity {



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // show action bar
        ActionBar myActionBar = getSupportActionBar();
        myActionBar.show();

        // Set the content of the activity to use the activity_main.xml layout file
        setContentView(R.layout.activity_lab);

        // Find the View that shows the sleep apnea category
        TextView sleepapnea = (TextView) findViewById(R.id.sleepapnea);

        // Set a click listener on that View
        sleepapnea.setOnClickListener((view) → {
                // Create a new intent to open the {@link SleepApnea}
                Intent sleepapneaIntent = new Intent( packageContext: LabActivity.this, SleepApnea.class);

                // Start the new activity
```

LabActivity

Gradle sync finished in 3 m 17 s 168 ms (7 minutes ago)

A2. Random Forest based temperature sensor characterization

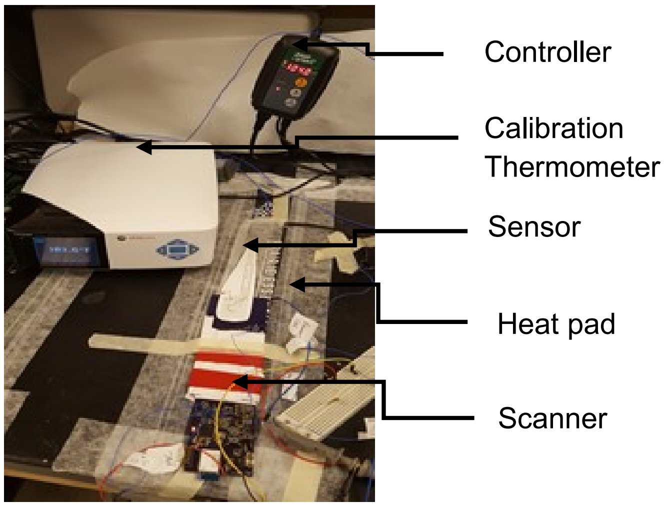


Fig. Experimental setup for temperature sensor characterization

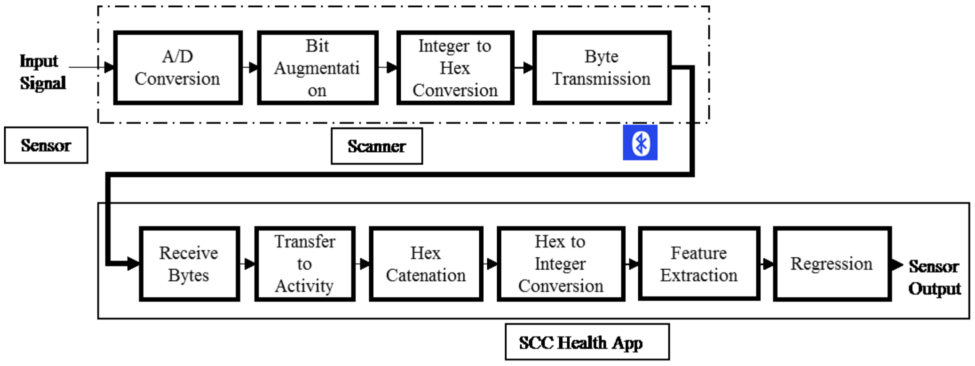
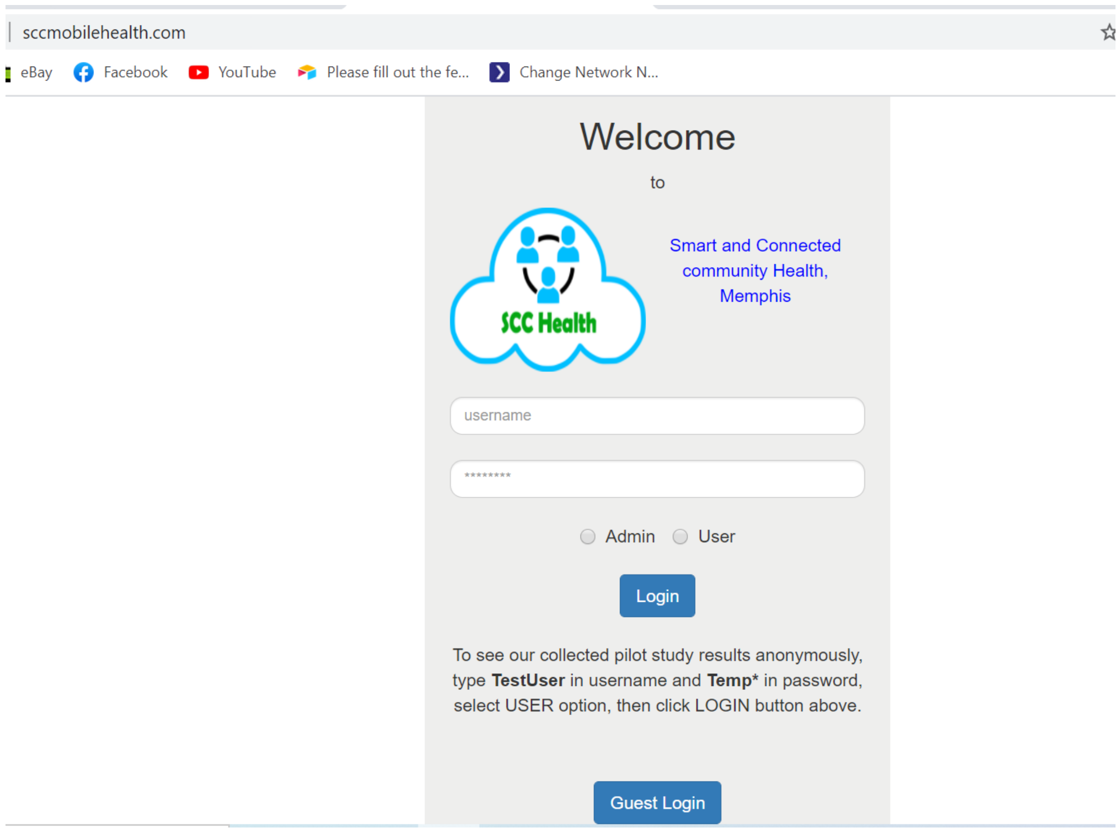


Fig. Block diagram for signal processing steps in the smartphone app

A3. The website showing spatiotemporal visualization of symptoms severity from the SCC-Health field study is available at www.sccmobilehealth.com. Snapshots from the webserver has been given below:

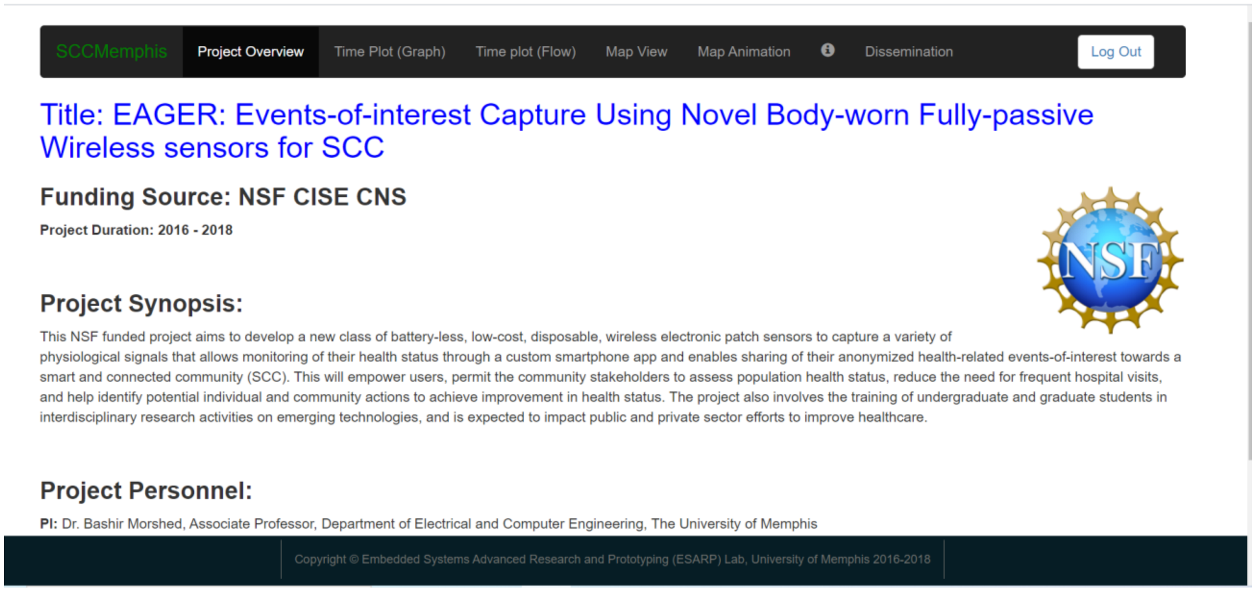


Welcome

to

Smart and Connected community Health, Memphis

To see our collected pilot study results anonymously, type **TestUser** in username and **Temp*** in password, select USER option, then click LOGIN button above.


SCCMemphis | Project Overview | Time Plot (Graph) | Time plot (Flow) | Map View | Map Animation | Dissemination | Log Out

Title: EAGER: Events-of-interest Capture Using Novel Body-worn Fully-passive Wireless sensors for SCC

Funding Source: NSF CISE CNS

Project Duration: 2016 - 2018

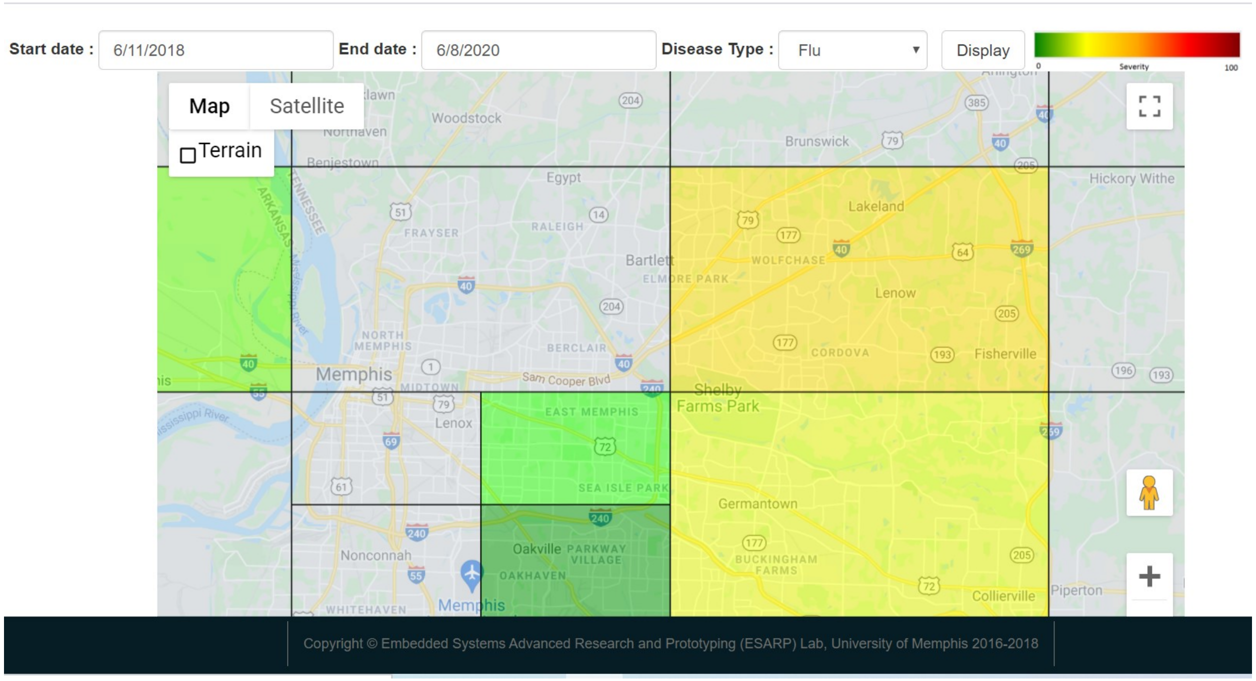
Project Synopsis:

This NSF funded project aims to develop a new class of battery-less, low-cost, disposable, wireless electronic patch sensors to capture a variety of physiological signals that allows monitoring of their health status through a custom smartphone app and enables sharing of their anonymized health-related events-of-interest towards a smart and connected community (SCC). This will empower users, permit the community stakeholders to assess population health status, reduce the need for frequent hospital visits, and help identify potential individual and community actions to achieve improvement in health status. The project also involves the training of undergraduate and graduate students in interdisciplinary research activities on emerging technologies, and is expected to impact public and private sector efforts to improve healthcare.

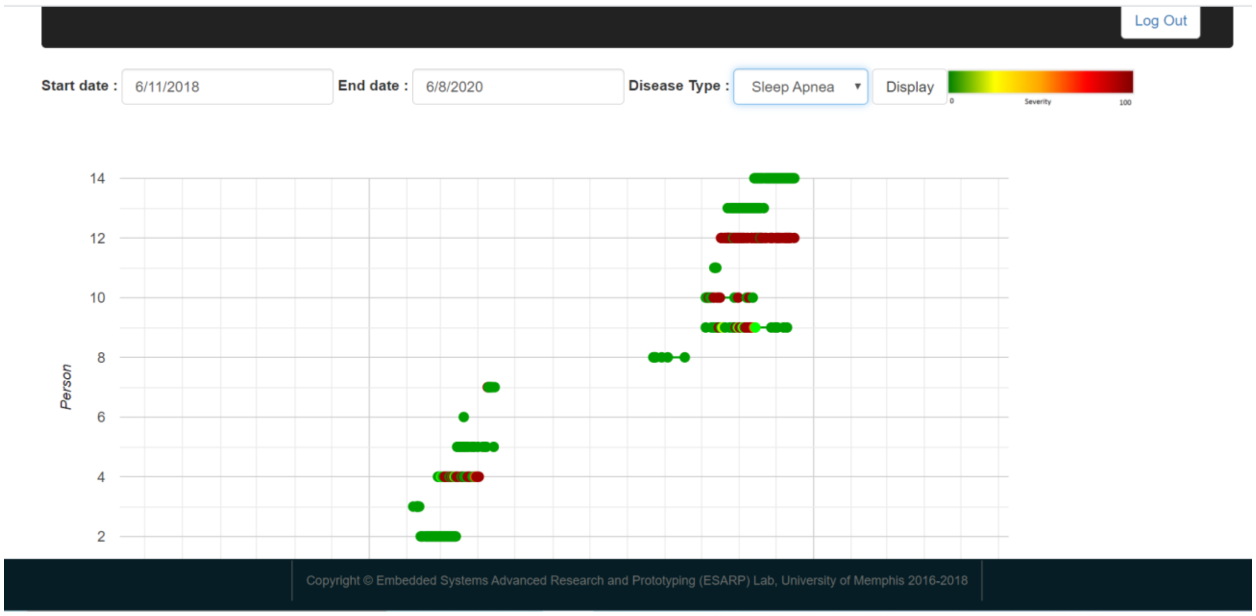
Project Personnel:

PI: Dr. Bashir Morshed, Associate Professor, Department of Electrical and Computer Engineering, The University of Memphis

Copyright © Embedded Systems Advanced Research and Prototyping (ESARP) Lab, University of Memphis 2016-2018


Start date : 6/11/2018 | End date : 6/8/2020 | Disease Type : Flu | Display

Copyright © Embedded Systems Advanced Research and Prototyping (ESARP) Lab, University of Memphis 2016-2018


Log Out
Start date :
6/11/2018
End date :
6/8/2020
Disease Type :
Sleep Apnea
Display
0
Severity
100
Person
14
12
10
8
6
4
2
Copyright © Embedded Systems Advanced Research and Prototyping (ESARP) Lab, University of Memphis 2016-2018

A4. Codes from the SCC- Health Application

```
package
nsf.esarplab.scchealth;


import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentTransaction;
import android.view.View;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;


public class LoginActivity extends FragmentActivity {
    Button btnSignIn, btnFragment;
    LoginDataBaseAdapter loginDataBaseAdapter;
    EditText ed1, ed2;
    private Context context;



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        Button b2=(Button)findViewById(R.id.button2);
        ed1=(EditText) findViewById(R.id.editText);


        // create a instance of SQLite Database
        loginDataBaseAdapter=new LoginDataBaseAdapter(this);
        loginDataBaseAdapter=loginDataBaseAdapter.open();


        // Get The Refference Of Buttons
        btnSignIn=(Button)findViewById(R.id.buttonSignIN);
        btnFragment=(Button)findViewById(R.id.help);


        //Set OnClick Listener on SignUp button
        btnFragment.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {


                FragmentManager fragmentManager = getSupportFragmentManager();
                FragmentTransaction fragmentTransaction =
fragmentManager.beginTransaction();
                HelpdeskFragment f1 = new HelpdeskFragment();
                fragmentTransaction.add(R.id.frag1, f1);
                fragmentTransaction.addToBackStack(null);
                fragmentTransaction.commit();
                // hide virtual keyboard
                InputMethodManager imm = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);
                imm.hideSoftInputFromWindow(ed1.getWindowToken(),
InputMethodManager.RESULT_UNCHANGED_SHOWN);
            }




        });
    }
    // Methos to handleClick Event of Sign In Button
    public void signIn(View V)
    {




        // get the References of views
        final  EditText editTextUserName=(EditText)findViewById(R.id.editText);
        final  EditText editTextPassword=(EditText)findViewById(R.id.editText2);
        // get The User name and Password
        String userName=editTextUserName.getText().toString();
        String password=editTextPassword.getText().toString();


        // fetch the Password form database for respective user name
        String storedPassword=loginDataBaseAdapter.getSinlgeEntry(userName);


        // check if the Stored password matches with  Password entered by user


        if ((userName.equals("admin") &&
password.equals("hce"))||(password.equals(storedPassword))) {


            // pass login details to shared preferences
            SharedPreferences prefs =
getSharedPreferences("logindetails",MODE_PRIVATE);
            SharedPreferences.Editor editor = prefs.edit();
            editor.putString("loginname",ed1.getText().toString()).commit();


            Intent welcomeIntent=new Intent(LoginActivity.this,WelcomeActivity.class);
            startActivity(welcomeIntent);


            Toast.makeText(LoginActivity.this, "Login Successful",
Toast.LENGTH_SHORT).show();


        } else {
            Toast.makeText(LoginActivity.this, "User Name or Password does not match",
Toast.LENGTH_LONG).show();
        }
    }




    @Override
    protected void onDestroy() {
        super.onDestroy();
        // Close The Database
        loginDataBaseAdapter.close();
    }




    public void onClick(View v) {
        finish();
    }
```

```
}

package
nsf.esarplab.scchealth;


import android.os.Bundle;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.widget.TextView;


public class About extends AppCompatActivity {


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_about);
        TextView project=(TextView) findViewById(R.id.about);
        //show actionbar
        ActionBar myActionBar = getSupportActionBar();
        myActionBar.show();


        String projectdescription="";
        projectdescription+="Project Title: EAGER: Events-of-interest Capture Using
Novel Body-worn " +
                "Fully-passive Wireless sensors for S&CC\n"+"\n";;
        projectdescription+="Funding Source: NSF CISE CNS\n  " +"\n";
        projectdescription+="Project Duration: 2016 - 2018\n " +"\n";
        projectdescription+="Project Synopsis:" +
                "\n" +
                "Patients with chronic illness require frequent and avoidable hospital visits.
" +
                "This project aims to develop a new class of battery-less, low-cost,
disposable, " +
                "wireless electronic patch sensors to monitor a variety of physiological
signals and " +
                "a custom smartphone app to monitor their health status and to elect to share
their" +
                " anonymized events-of-interest with their community towards a smart and
connected " +
                "community (S&CC). This will empower users, permit the community
stakeholders to assess" +
                " population health status, reduce the need for frequent hospital visits, and
help " +
                "identify potential individual and community actions to achieve
improvement in health" +
                " status. The project also involves the training of undergraduate and
graduate students" +
                " in interdisciplinary research activities on emerging technologies, and is
expected to" +
                " impact public and private sector efforts to improve healthcare.\n" +
                "\n" +
                "PI: Dr. Bashir Morshed, Associate Professor, Department of Electrical and
Computer " +
                "Engineering, The University of Memphis\n" +
                "\n" +
                "Co-PI: Dr. Brook Harmon, Assistant Professor, School of Public Health,
The University of Memphis"+"\n\n"
                +"Consultant: Dr. M. Rahman,  Baptist Minor Medical Center, Memphis,
TN"+"\n\n"
                +"Collaborator: Memphis District of The United Methodist Church
(UMC)";
        project.setText(projectdescription);



    }
}
```

```
package
nsf.esarplab.
scchealth;


import android.bluetooth.BluetoothAdapter;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.net.Uri;
import android.net.wifi.WifiManager;
import android.os.Bundle;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.text.SpannableString;
import android.text.style.RelativeSizeSpan;
import android.util.Log;
import android.view.View;
import android.widget.CompoundButton;
import android.widget.Switch;
import android.widget.TextView;


import static nsf.esarplab.bluetoothlibrary.BluetoothState.REQUEST_ENABLE_BT;



public class HomeActivity extends AppCompatActivity {
  final BluetoothAdapter bluetooth = BluetoothAdapter.getDefaultAdapter();
  private Switch btSwitch;
  private WifiManager wifiManager;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //getActionBar().setIcon(new
ColorDrawable(getResources().getColor(android.R.color.transparent)));
        // Set the content of the activity to use the activity_main.xml layout file
        setContentView(R.layout.activity_home);


        ActionBar myActionBar = getSupportActionBar();
        myActionBar.show();


        TextView intro = (TextView) findViewById(R.id.espeech);
        intro.setVisibility(View.INVISIBLE);
        Log.i("Home", "Called");


        // Find the switch that turn on/off bluetooth
        btSwitch = (Switch) findViewById(R.id.mySwitch);


        //Get Login details
        SharedPreferences prefs = getSharedPreferences("logindetails",MODE_PRIVATE);
        String Uname =  prefs.getString("loginname","Default");




        // Find the switch that turn on/off wifi
        //wifiSwitch = (Switch) findViewById(R.id.wifiSwitch);




        // Find the View that shows the  project information
        TextView about = (TextView) findViewById(R.id.about);
        String aboutString = "About \n\t- Know the project";
        SpannableString ss1 = new SpannableString(aboutString);
        ss1.setSpan(new RelativeSizeSpan(2.0f), 0, 5, 0); // set size
        about.setText(ss1);
        // Set a click listener on that View


        about.setOnClickListener(new View.OnClickListener() {
            // The code in this method will be executed when the lab category is clicked on.
            @Override
            public void onClick(View view) {
                // Create a new intent to open the {@link LabActivity}
                Intent aboutIntent = new Intent(HomeActivity.this, About.class);


                // Start the new activity
                startActivity(aboutIntent);
    }
});


// Find the View that shows the lab category
TextView lab = (TextView) findViewById(R.id.lab);
String diagnosticString = "Diagnostic \n\t- Diagnose flu, arrythmia, sleep apnea & COPD";
SpannableString ss3 = new SpannableString(diagnosticString);
ss3.setSpan(new RelativeSizeSpan(2.0f), 0, 10, 0); // set size
lab.setText(ss3);
// Set a click listener on that View


lab.setOnClickListener(new View.OnClickListener() {
    // The code in this method will be executed when the lab category is clicked on.
    @Override
    public void onClick(View view) {
        // Create a new intent to open the {@link LabActivity}
        Intent labIntent = new Intent(HomeActivity.this, LabActivity.class);


        // Start the new activity
        startActivity(labIntent);


    }
});




// Find the View that shows the setting category
TextView setting = (TextView) findViewById(R.id.setting);
String settingString = "Settings \n\t- Create user, manage profiles & setup network";
SpannableString ss2 = new SpannableString(settingString);
ss2.setSpan(new RelativeSizeSpan(2.0f), 0, 8, 0); // set size
setting.setText(ss2);
// Set a click listener on that View
setting.setOnClickListener(new View.OnClickListener() {
    // The code in this method will be executed when the profile category is clicked on.
    @Override
    public void onClick(View view) {
        // Create a new intent to open the {@link ProfileActivity}
        Intent settingIntent = new Intent(HomeActivity.this, DB_login.class);


        // Start the new activity
        startActivity(settingIntent);


    }
});


// Find the View that shows the website category
```

```
        TextView web = (TextView) findViewById(R.id.web);
        String webString = "Website \n\t- Visit SCC Health Website";
        SpannableString ss4 = new SpannableString(webString);
        ss4.setSpan(new RelativeSizeSpan(2.0f), 0, 8, 0); // set size
        web.setText(ss4);
        // Set a click listener on that View
        web.setOnClickListener(new View.OnClickListener() {
            // The code in this method will be executed when the profile category is clicked on.
            @Override
            public void onClick(View view) {
                Uri uri = Uri.parse("http://sscmemphis.com"); // missing 'http://' will cause crashed
                Intent intent = new Intent(Intent.ACTION_VIEW, uri);
                startActivity(intent);
            }
        });


        // manage switch position based on connection status
        if (bluetooth.isEnabled()){
            btSwitch.setChecked(true);
        }else {
            btSwitch.setChecked(false);
        }







        //attach a listener to check for changes in state
        btSwitch.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {


            @Override
            public void onCheckedChanged(CompoundButton buttonView,
                                         boolean isChecked) {


                if (isChecked) {
                    if (!bluetooth.isEnabled()) {
                        // prompt the user to turn BlueTooth on
                        Intent enableBtIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                        startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
                    }
                } else {
                    bluetooth.disable();
                }


            }
        });
        // put setting not clickable
        if (!(Uname.matches("admin"))){
            setting.setClickable(false);
            setting.setBackgroundColor(Color.GRAY);
        }
        // manage switch to turn wifi on/off


       /* wifiManager = (WifiManager)
this.getApplicationContext().getSystemService(Context.WIFI_SERVICE);
        if(wifiManager.isWifiEnabled()){
            wifiSwitch.setChecked(true);
        }else{
            wifiSwitch.setChecked(false);

        }

        //attach a listener to check for changes in wifi state
        wifiSwitch.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {

            @Override
            public void onCheckedChanged(CompoundButton buttonView,
                                         boolean isChecked) {

                if (isChecked) {
                    wifiManager.setWifiEnabled(true);
                } else {
                    wifiManager.setWifiEnabled(false);
                }

            }
        });*/



    }





    public void logOut(View v) {
        // close the app
        try {


            Intent intentBack = new Intent(this,LoginActivity.class);
            startActivity(intentBack);
            Intent intentExit = new Intent(Intent.ACTION_MAIN);
            intentExit.addCategory(Intent.CATEGORY_HOME);
            intentExit.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            startActivity(intentExit);
            finish();




        } catch (Exception e) {
            e.printStackTrace();
        }


    }
}


package nsf.esarplab.scchealth;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.os.Environment;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.text.method.ScrollingMovementMethod;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.PopupWindow;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.common.api.GoogleApiClient;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Calendar;

import nsf.esarplab.bluetoothlibrary.BluetoothSPP;
import nsf.esarplab.bluetoothlibrary.BluetoothSPP.BluetoothConnectionListener;
import nsf.esarplab.bluetoothlibrary.BluetoothState;
import nsf.esarplab.bluetoothlibrary.DeviceList;

public class SleepApnea extends AppCompatActivity {

    BluetoothSPP bt;
    Button test, csvReader;
    TextView textReceived, connectionRead;
    EditText etMessage;
```

```
private GoogleApiClient client;
private boolean saReceived = false;
private boolean hrReceived = false;
private boolean tenReceived = false;
private boolean btdata = true;
Menu menu;
String s = "";

@Override
protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_sleep_apnea);
    bt = new BluetoothSPP(this);
    // show actionbar
    ActionBar myActionBar = getSupportActionBar();
    myActionBar.show();

    // Set active profile

    TextView mNameText = (TextView) findViewById(R.id.display_name);

    //Show active profile
    SharedPreferences prefs = getSharedPreferences("logindetails",MODE_PRIVATE);
    String Uname =  prefs.getString("loginname","Default");
    mNameText.setText("\t\t"+Uname);

    s = mNameText.getText().toString().trim();

    /*//reading profile from file
    try {
        FileInputStream fileIn = openFileInput("mytextfile.txt");
        InputStreamReader InputRead = new InputStreamReader(fileIn);
        char[] inputBuffer = new char[READ_BLOCK_SIZE];
        *//*String s="";*//*
        int charRead;
        while ((charRead = InputRead.read(inputBuffer)) > 0) {
            // char to string conversion
            String readstring = String.copyValueOf(inputBuffer, 0, charRead);
            s += readstring;
        }
        InputRead.close();
        *//*mNameText.setText(s);*//*
        *//*Toast.makeText(getBaseContext(), s,Toast.LENGTH_SHORT).show();*//*
    } catch (Exception e) {
        e.printStackTrace();
    }
    mNameText.setText(s);*/

    // set views

    test=(Button) findViewById(R.id.test);
    csvReader=(Button) findViewById(R.id.readCSV);
    textReceived = (TextView) findViewById(R.id.display_name);
    connectionRead = (TextView) findViewById(R.id.textStatus);
    textReceived.setMovementMethod(new ScrollingMovementMethod());
```

```
bt.setOnDataReceivedListener(new BluetoothSPP.OnDataReceivedListener() {
   public void onDataReceived(byte[] data, String message) {

      textReceived.append(message + "\n");
      if (btdata) {
         try {
            writeToCsv(message);
         } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
         }
      }

      if (tenReceived == true) {
         //receive data

         textReceived.append(message + "\n");
                              /*try {
                                 writeToCsv(message);
                              } catch (IOException e) {
                                 // TODO Auto-generated catch block
                                 e.printStackTrace();
                              }*/
      } else {
         if (message.equals("SA")) {
            bt.send("HR", true);
            saReceived = true;
         } else {
            if (message.equals("HR") && saReceived) {
               bt.send("10", true);
               hrReceived = true;
            } else {
               if (message.equals("10") && saReceived && hrReceived) {
                  bt.send("OK", true);
                  tenReceived = true;
               } else {
                  textReceived.append("Failed Handshake");
               }
            }
         }
      }
   }

});


bt.setBluetoothConnectionListener(new BluetoothConnectionListener() {
   public void onDeviceDisconnected() {
      connectionRead.setText("Status : Not connect");
      menu.clear();
      getMenuInflater().inflate(R.menu.menu_connection, menu);
   }

   public void onDeviceConnectionFailed() {
        connectionRead.setText("Status : Connection failed");
    }

    public void onDeviceConnected(String name, String address) {
        connectionRead.setText("Status : Connected to " + name);
        menu.clear();
        getMenuInflater().inflate(R.menu.menu_disconnection, menu);
    }
});

/*IntentFilter filter = new IntentFilter();
filter.addAction("SOME_ACTION");
filter.addAction("SOME_OTHER_ACTION");
BroadcastReceiver receiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        TextView display = (TextView) findViewById(R.id.display_name);
        display.append(action);
        *//*Log.i("Receiver", "Broadcast received: " + action);
        if (action.equals("my.action.string")) {
            String state = intent.getExtras().getString("extra");
            display.append(state);
        }*//*
    }
};
registerReceiver(receiver, filter);*/


final TextView btnOpenPopup = (TextView) findViewById(R.id.info);
btnOpenPopup.setOnClickListener(new Button.OnClickListener() {

    @Override
    public void onClick(View arg0) {
        LayoutInflater layoutInflater
                = (LayoutInflater) getBaseContext()
                .getSystemService(LAYOUT_INFLATER_SERVICE);
        View popupView = layoutInflater.inflate(R.layout.info_sleepapnea, null);
        final PopupWindow popupWindow = new PopupWindow(
                popupView,
                ViewGroup.LayoutParams.WRAP_CONTENT,
                ViewGroup.LayoutParams.WRAP_CONTENT);

        Button btnDismiss = (Button) popupView.findViewById(R.id.dismiss);
        btnDismiss.setOnClickListener(new Button.OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                popupWindow.dismiss();
            }
        });

        popupWindow.showAsDropDown(btnOpenPopup, 50, -30);

    }
    });

    final TextView btnOpenInstruction = (TextView) findViewById(R.id.inst);
    btnOpenInstruction.setOnClickListener(new Button.OnClickListener() {

        @Override
        public void onClick(View arg0) {
            LayoutInflater layoutInflater
                    = (LayoutInflater) getBaseContext()
                    .getSystemService(LAYOUT_INFLATER_SERVICE);
            View popupView = layoutInflater.inflate(R.layout.inst_sleepapnea, null);
            final PopupWindow popupWindow = new PopupWindow(
                    popupView,
                    ViewGroup.LayoutParams.WRAP_CONTENT,
                    ViewGroup.LayoutParams.WRAP_CONTENT);

            Button btnDismiss = (Button) popupView.findViewById(R.id.dismiss);
            btnDismiss.setOnClickListener(new Button.OnClickListener() {

                @Override
                public void onClick(View v) {
                    // TODO Auto-generated method stub
                    popupWindow.dismiss();
                }
            });

            popupWindow.showAsDropDown(btnOpenPopup, 50, -30);

        }
    });


}

public void onDestroy() {
    super.onDestroy();
    bt.stopService();
}

public boolean onCreateOptionsMenu(Menu menu) {
    this.menu = menu;
    getMenuInflater().inflate(R.menu.menu_connection, menu);
    return true;
}

public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    if (id == R.id.menu_android_connect) {
        bt.setDeviceTarget(BluetoothState.DEVICE_ANDROID);
        /*
                        if(bt.getServiceState() == BluetoothState.STATE_CONNECTED)
                        bt.disconnect();*/
        Intent intent = new Intent(getApplicationContext(), DeviceList.class);
        startActivityForResult(intent, BluetoothState.REQUEST_CONNECT_DEVICE);
    } else if (id == R.id.menu_device_connect) {
      bt.setDeviceTarget(BluetoothState.DEVICE_OTHER);
                              /*
                              if(bt.getServiceState() == BluetoothState.STATE_CONNECTED)
                              bt.disconnect();*/
      Intent intent = new Intent(getApplicationContext(), DeviceList.class);
      startActivityForResult(intent, BluetoothState.REQUEST_CONNECT_DEVICE);
    } else if (id == R.id.menu_disconnect) {
      if (bt.getServiceState() == BluetoothState.STATE_CONNECTED)
        bt.disconnect();
    }

    else if (id == R.id.menu_reinitialize) {
      textReceived.setText("");
    }
    return super.onOptionsItemSelected(item);
  }
  public void onStart() {
    super.onStart();
    if (!bt.isBluetoothEnabled()) {
      Intent intent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
      startActivityForResult(intent, BluetoothState.REQUEST_ENABLE_BT);
    } else {
      if (!bt.isServiceAvailable()) {
        bt.setupService();
        bt.startService(BluetoothState.DEVICE_ANDROID);
        setup();
        connectScanner();

      }
    }

  }
  public void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == BluetoothState.REQUEST_CONNECT_DEVICE) {
      if (resultCode == Activity.RESULT_OK)
        bt.connect(data);
    } else if (requestCode == BluetoothState.REQUEST_ENABLE_BT) {
      if (resultCode == Activity.RESULT_OK) {
        bt.setupService();
        bt.startService(BluetoothState.DEVICE_ANDROID);
        setup();
      } else {
        Toast.makeText(getApplicationContext()
             , "BluetoothActivity was not enabled."
             , Toast.LENGTH_SHORT).show();
        finish();
      }
    }
  }

  public void setup() {

    test.setOnClickListener(new View.OnClickListener() {
      public void onClick(View v) {
        {
```

```
                    bt.send("SA", true);


                }
            }
        });
    }

    public void connectScanner() {

        csvReader.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {


            }
        });
    }

    //write to csv file
    public void writeToCsv(String x) throws IOException{
        Calendar c = Calendar.getInstance();
        File folder = new File(Environment.getExternalStorageDirectory() + "/project");
        boolean success = true;
        if (!folder.exists()) {
            success = folder.mkdir();
        }
        if (success) {
            // Do something on success
            String csv = "/storage/sdcard0/project/btvalue.csv";
            FileWriter file_writer = new FileWriter(csv,true);;



            String s=
c.get(Calendar.YEAR)+","+(c.get(Calendar.MONTH)+1)+","+c.get(Calendar.DATE)+","+c.ge
t(Calendar.HOUR)+","+c.get(Calendar.MINUTE)+","+c.get(Calendar.SECOND)+","+
c.get(Calendar.MILLISECOND)+","+x + "\n";

            file_writer.append(s);
            file_writer.close();

        }



    }
    @Override
    public void onStop() {
        super.onStop();

    }
}
```
package
nsf.esarp

```
lab.sccheh
alth;

                  import android.app.Activity;
                  import android.bluetooth.BluetoothAdapter;
                  import android.content.Intent;
                  import android.content.SharedPreferences;
                  import android.os.Bundle;
                  import android.os.Environment;
                  import android.support.v7.app.ActionBar;
                  import android.support.v7.app.AppCompatActivity;
                  import android.text.method.ScrollingMovementMethod;
                  import android.view.LayoutInflater;
                  import android.view.Menu;
                  import android.view.MenuItem;
                  import android.view.View;
                  import android.view.ViewGroup;
                  import android.widget.Button;
                  import android.widget.EditText;
                  import android.widget.PopupWindow;
                  import android.widget.TextView;
                  import android.widget.Toast;


                  import java.io.File;
                  import java.io.FileWriter;
                  import java.io.IOException;
                  import java.util.Calendar;


                  import nsf.esarplab.bluetoothlibrary.BluetoothSPP;
                  import nsf.esarplab.bluetoothlibrary.BluetoothState;
                  import nsf.esarplab.bluetoothlibrary.DeviceList;


                  //import com.google.android.gms.common.api.GoogleApiClient;


                  public class ArrhythmiaActivity extends AppCompatActivity {
                     BluetoothSPP bt;
                     Button test, csvReader;
                     TextView textReceived, connectionRead;
                     EditText etMessage;
                     //private GoogleApiClient client;
                     private boolean saReceived = false;
                     private boolean hrReceived = false;
                     private boolean tenReceived = false;
                     private boolean btdata = true;
                     Menu menu;
                     String s = "";


                     @Override
                     protected void onCreate(Bundle savedInstanceState) {
```

```
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_arrythmia);
bt = new BluetoothSPP(this);
// show actionbar
ActionBar myActionBar = getSupportActionBar();
myActionBar.show();
TextView mNameText = (TextView) findViewById(R.id.display_name);


//Show active profile
SharedPreferences prefs = getSharedPreferences("logindetails",MODE_PRIVATE);
String Uname =  prefs.getString("loginname","Default");
mNameText.setText("\t\t"+Uname);


s = mNameText.getText().toString().trim();






/*//reading profile from file
try {
   FileInputStream fileIn = openFileInput("mytextfile.txt");
   InputStreamReader InputRead = new InputStreamReader(fileIn);
   char[] inputBuffer = new char[READ_BLOCK_SIZE];
   *//*String s="";*//*
   int charRead;

   while ((charRead = InputRead.read(inputBuffer)) > 0) {
      // char to string conversion
      String readstring = String.copyValueOf(inputBuffer, 0, charRead);
      s += readstring;
   }
   InputRead.close();
   *//*mNameText.setText(s);*//*
   *//*Toast.makeText(getBaseContext(), s,Toast.LENGTH_SHORT).show();*//*

} catch (Exception e) {
   e.printStackTrace();
}
mNameText.setText(s);*/


// receive intent
test=(Button) findViewById(R.id.test);
csvReader=(Button) findViewById(R.id.readCSV);
textReceived = (TextView) findViewById(R.id.display_name);
connectionRead = (TextView) findViewById(R.id.textStatus);
textReceived.setMovementMethod(new ScrollingMovementMethod());
```

```
bt.setOnDataReceivedListener(new BluetoothSPP.OnDataReceivedListener() {
  public void onDataReceived(byte[] data, String message) {


    textReceived.append(message + "\n");
    if (btdata) {
      try {
        writeToCsv(message);
      } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
      }
    }


    if (tenReceived == true) {
      //receive data


      textReceived.append(message + "\n");
                    /*try {
                      writeToCsv(message);
                    } catch (IOException e) {
                      // TODO Auto-generated catch block
                      e.printStackTrace();
                    }*/
    } else {
      if (message.equals("SA")) {
        bt.send("HR", true);
        saReceived = true;
      } else {
        if (message.equals("HR") && saReceived) {
          bt.send("10", true);
          hrReceived = true;
        } else {
          if (message.equals("10") && saReceived && hrReceived) {
            bt.send("OK", true);
            tenReceived = true;
          } else {
            textReceived.append("Failed Handshake");
          }
        }
      }
    }
  }


});




bt.setBluetoothConnectionListener(new BluetoothSPP.BluetoothConnectionListener() {
  public void onDeviceDisconnected() {
    connectionRead.setText("Status : Not connect");
    menu.clear();
```

```
        getMenuInflater().inflate(R.menu.menu_connection, menu);
    }


    public void onDeviceConnectionFailed() {
        connectionRead.setText("Status : Connection failed");
    }


    public void onDeviceConnected(String name, String address) {
        connectionRead.setText("Status : Connected to " + name);
        menu.clear();
        getMenuInflater().inflate(R.menu.menu_disconnection, menu);
    }
});


/*IntentFilter filter = new IntentFilter();
filter.addAction("SOME_ACTION");
filter.addAction("SOME_OTHER_ACTION");

BroadcastReceiver receiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        TextView display = (TextView) findViewById(R.id.display_name);
        display.append(action);
        *//*Log.i("Receiver", "Broadcast received: " + action);

        if (action.equals("my.action.string")) {
            String state = intent.getExtras().getString("extra");
            display.append(state);
        }*//*
    }

};
registerReceiver(receiver, filter);*/




final TextView btnOpenPopup = (TextView) findViewById(R.id.info);
btnOpenPopup.setOnClickListener(new Button.OnClickListener() {


    @Override
    public void onClick(View arg0) {
        LayoutInflater layoutInflater
                = (LayoutInflater) getBaseContext()
                .getSystemService(LAYOUT_INFLATER_SERVICE);
        View popupView = layoutInflater.inflate(R.layout.info_arrhythmia, null);
        final PopupWindow popupWindow = new PopupWindow(
                popupView,
                ViewGroup.LayoutParams.WRAP_CONTENT,
                ViewGroup.LayoutParams.WRAP_CONTENT);
        Button btnDismiss = (Button) popupView.findViewById(R.id.dismiss);
        btnDismiss.setOnClickListener(new Button.OnClickListener() {


            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                popupWindow.dismiss();
            }
        });


        popupWindow.showAsDropDown(btnOpenPopup, 50, -30);


    }
});


final TextView btnOpenInstruction = (TextView) findViewById(R.id.inst);
btnOpenInstruction.setOnClickListener(new Button.OnClickListener() {


    @Override
    public void onClick(View arg0) {
        LayoutInflater layoutInflater
                = (LayoutInflater) getBaseContext()
                .getSystemService(LAYOUT_INFLATER_SERVICE);
        View popupView = layoutInflater.inflate(R.layout.inst_arrhythmia, null);
        final PopupWindow popupWindow = new PopupWindow(
                popupView,
                ViewGroup.LayoutParams.WRAP_CONTENT,
                ViewGroup.LayoutParams.WRAP_CONTENT);


        Button btnDismiss = (Button) popupView.findViewById(R.id.dismiss);
        btnDismiss.setOnClickListener(new Button.OnClickListener() {


            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                popupWindow.dismiss();
            }
        });


        popupWindow.showAsDropDown(btnOpenPopup, 50, -30);


    }
});
```

```
}


public void onDestroy() {
    super.onDestroy();
    bt.stopService();
}


public boolean onCreateOptionsMenu(Menu menu) {
    this.menu = menu;
    getMenuInflater().inflate(R.menu.menu_connection, menu);
    return true;
}


public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    if (id == R.id.menu_android_connect) {
        bt.setDeviceTarget(BluetoothState.DEVICE_ANDROID);
        /*
                                if(bt.getServiceState() == BluetoothState.STATE_CONNECTED)
                                bt.disconnect();*/
        Intent intent = new Intent(getApplicationContext(), DeviceList.class);
        startActivityForResult(intent, BluetoothState.REQUEST_CONNECT_DEVICE);




    } else if (id == R.id.menu_device_connect) {
        bt.setDeviceTarget(BluetoothState.DEVICE_OTHER);
                                /*
                                if(bt.getServiceState() == BluetoothState.STATE_CONNECTED)
                                bt.disconnect();*/
        Intent intent = new Intent(getApplicationContext(), DeviceList.class);
        startActivityForResult(intent, BluetoothState.REQUEST_CONNECT_DEVICE);
    } else if (id == R.id.menu_disconnect) {
        if (bt.getServiceState() == BluetoothState.STATE_CONNECTED)
            bt.disconnect();
    }


    else if (id == R.id.menu_reinitialize) {
        textReceived.setText("");
    }
    return super.onOptionsItemSelected(item);
}
public void onStart() {
    super.onStart();
    if (!bt.isBluetoothEnabled()) {
        Intent intent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(intent, BluetoothState.REQUEST_ENABLE_BT);
    } else {
        if (!bt.isServiceAvailable()) {
            bt.setupService();
            bt.startService(BluetoothState.DEVICE_ANDROID);
            setup();
            connectScanner();


        }
    }


}
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == BluetoothState.REQUEST_CONNECT_DEVICE) {
        if (resultCode == Activity.RESULT_OK)
            bt.connect(data);
    } else if (requestCode == BluetoothState.REQUEST_ENABLE_BT) {
        if (resultCode == Activity.RESULT_OK) {
            bt.setupService();
            bt.startService(BluetoothState.DEVICE_ANDROID);
            setup();
        } else {
            Toast.makeText(getApplicationContext()
                    , "BluetoothActivity was not enabled."
                    , Toast.LENGTH_SHORT).show();
            finish();
        }
    }
}


public void setup() {


    test.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            {
                bt.send("SA", true);




            }
        }
    });
}


public void connectScanner() {


    csvReader.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {




        }
        });
    }


    //write to csv file
    public void writeToCsv(String x) throws IOException {
        Calendar c = Calendar.getInstance();
        File folder = new File(Environment.getExternalStorageDirectory() + "/project");
        boolean success = true;
        if (!folder.exists()) {
            success = folder.mkdir();
        }
        if (success) {
            // Do something on success
            String csv = "/storage/sdcard0/project/btvalue.csv";
            FileWriter file_writer = new FileWriter(csv,true);;









            String s=
c.get(Calendar.YEAR)+","+(c.get(Calendar.MONTH)+1)+","+c.get(Calendar.DATE)+","+c.get(C
alendar.HOUR)+","+c.get(Calendar.MINUTE)+","+c.get(Calendar.SECOND)+","+
c.get(Calendar.MILLISECOND)+","+x + "\n";


            file_writer.append(s);
            file_writer.close();


        }






    }
    @Override
    public void onStop() {
        super.onStop();


    }
}
package
nsf.esarplab.scc
health;


            import android.app.Activity;
            import android.bluetooth.BluetoothAdapter;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.PopupWindow;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import android.widget.Toast;


//import com.google.android.gms.common.api.GoogleApiClient;


import nsf.esarplab.bluetoothlibrary.BluetoothSPP;
import nsf.esarplab.bluetoothlibrary.BluetoothState;
import nsf.esarplab.bluetoothlibrary.DeviceList;


public class AsthmaActivity extends AppCompatActivity {
   BluetoothSPP bt;
   Button test;
   TextView hrData, oximetryData, connectionRead;
   EditText etMessage;
   //private GoogleApiClient client;
   private boolean pdReceived = false;
   private boolean hrReceived = false;
   private boolean tenReceived = false;
   private boolean poReceived=false;
   private int sensor=1;
   Menu menu;
   String s = "";


   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_asthma);


      // show action bar
      ActionBar myActionBar = getSupportActionBar();
      myActionBar.show();
      TextView mNameText = (TextView) findViewById(R.id.display_name);
      bt = new BluetoothSPP(this);
      // receive intent
      test=(Button) findViewById(R.id.test);
```

```
hrData = (TextView) findViewById(R.id.display_bdt);
oximetryData=(TextView) findViewById(R.id.display_spo2);
connectionRead = (TextView) findViewById(R.id.textStatus);




//Show active profile
SharedPreferences prefs = getSharedPreferences("logindetails",MODE_PRIVATE);
String Uname =  prefs.getString("loginname","Default");
mNameText.setText("\t\t"+Uname);


s = mNameText.getText().toString().trim();
// Set active profile




/*//reading profile from file
try {
    FileInputStream fileIn = openFileInput("mytextfile.txt");
    InputStreamReader InputRead = new InputStreamReader(fileIn);
    char[] inputBuffer = new char[READ_BLOCK_SIZE];
    *//*String s="";*//*
    int charRead;

    while ((charRead = InputRead.read(inputBuffer)) > 0) {
        // char to string conversion
        String readstring = String.copyValueOf(inputBuffer, 0, charRead);
        s += readstring;
    }
    InputRead.close();
    *//*mNameText.setText(s);*//*
    *//*Toast.makeText(getBaseContext(), s,Toast.LENGTH_SHORT).show();*//*

} catch (Exception e) {
    e.printStackTrace();
}
mNameText.setText(s);*/


// set the bluetooth connection


bt.setOnDataReceivedListener(new BluetoothSPP.OnDataReceivedListener() {
    public void onDataReceived(byte[] data, String message) {


        //textReceived.append(message + "\n");
        if (tenReceived == true) {
            //receive data
            switch (sensor) {
                case 1: {
                    hrData.append(message + "\n");
                    break;
                }
                case 2: {


                    oximetryData.append(message + "\n");
                    break;
                }


            }


        } else {
            if (message.equals("PD")) {


                switch (sensor) {
                    case 1: {
                        bt.send("HR", true);
                        pdReceived = true;
                        break;
                    }
                    case 2: {
                        bt.send("PO", true);
                        pdReceived = true;
                        break;
                    }
                }
            } else {
                if (message.equals("HR") && pdReceived) {
                    bt.send("10", true);
                    hrReceived = true;
                } else if (message.equals("PO") && pdReceived) {
                    bt.send("5", true);
                    poReceived = true;
                } else {
                    if (message.equals("10") && pdReceived && hrReceived) {
                        bt.send("OK", true);
                        tenReceived = true;
                    }
                    else if (message.equals("5") && pdReceived && poReceived) {
                        bt.send("OK", true);
                        tenReceived = true;
                    } else {
                        hrData.append("Failed Handshake");
                    }
                }
            }
        }
    }


});
```

```
        bt.setBluetoothConnectionListener(new BluetoothSPP.BluetoothConnectionListener()
{
            public void onDeviceDisconnected() {
                connectionRead.setText("Status : Not connect");
                menu.clear();
                getMenuInflater().inflate(R.menu.menu_connection, menu);
            }


            public void onDeviceConnectionFailed() {
                connectionRead.setText("Status : Connection failed");
            }


            public void onDeviceConnected(String name, String address) {
                connectionRead.setText("Status : Connected to " + name);
                menu.clear();
                getMenuInflater().inflate(R.menu.menu_disconnection, menu);
            }
        });





        // set corresponding display for selected sensor
        final LinearLayout hrlayout = (LinearLayout) findViewById(R.id.hr_result);
        final LinearLayout spo2layout = (LinearLayout) findViewById(R.id.spo2_result);
        RadioButton rbOxygen = (RadioButton) findViewById(R.id.rb_o2);
        RadioButton rbheartRate = (RadioButton) findViewById(R.id.rb_hr);
        hrlayout.setVisibility(View.VISIBLE);
        spo2layout.setVisibility(View.GONE);


        RadioGroup radioGroup = (RadioGroup) findViewById(R.id.rg1);
        radioGroup.setOnCheckedChangeListener(new
RadioGroup.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(RadioGroup group, int checkedId) {
                if (checkedId == R.id.rb_o2) {
                    sensor=2;
                    hrlayout.setVisibility(View.GONE);
                    spo2layout.setVisibility(View.VISIBLE);
                } else {
                    hrlayout.setVisibility(View.VISIBLE);
                    spo2layout.setVisibility(View.GONE);
                }
            }
        });


// inflate the screen for info and instruction
        final TextView btnOpenPopup = (TextView) findViewById(R.id.info);
        btnOpenPopup.setOnClickListener(new Button.OnClickListener() {
```

```
  @Override
  public void onClick(View arg0) {
    LayoutInflater layoutInflater
        = (LayoutInflater) getBaseContext()
        .getSystemService(LAYOUT_INFLATER_SERVICE);
    View popupView = layoutInflater.inflate(R.layout.info_copd, null);
    final PopupWindow popupWindow = new PopupWindow(
        popupView,
        ViewGroup.LayoutParams.WRAP_CONTENT,
        ViewGroup.LayoutParams.WRAP_CONTENT);


    Button btnDismiss = (Button) popupView.findViewById(R.id.dismiss);
    btnDismiss.setOnClickListener(new Button.OnClickListener() {


      @Override
      public void onClick(View v) {
        // TODO Auto-generated method stub
        popupWindow.dismiss();
      }
    });


    popupWindow.showAsDropDown(btnOpenPopup, 50, -30);


  }
});


final TextView btnOpenInstruction = (TextView) findViewById(R.id.inst);
btnOpenInstruction.setOnClickListener(new Button.OnClickListener() {


  @Override
  public void onClick(View arg0) {
    LayoutInflater layoutInflater
        = (LayoutInflater) getBaseContext()
        .getSystemService(LAYOUT_INFLATER_SERVICE);
    View popupView = layoutInflater.inflate(R.layout.inst_copd, null);
    final PopupWindow popupWindow = new PopupWindow(
        popupView,
        ViewGroup.LayoutParams.WRAP_CONTENT,
        ViewGroup.LayoutParams.WRAP_CONTENT);


    Button btnDismiss = (Button) popupView.findViewById(R.id.dismiss);
    btnDismiss.setOnClickListener(new Button.OnClickListener() {


      @Override
      public void onClick(View v) {
        // TODO Auto-generated method stub
        popupWindow.dismiss();
```

```
                }
            });


            popupWindow.showAsDropDown(btnOpenPopup, 50, -30);


        }
    });
}


// @Override
/* public void onStart() {
    super.onStart();
    if (!mBoundService.isBluetoothEnabled()) {
        Intent intent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(intent, BluetoothState.REQUEST_ENABLE_BT);
    } else {
        if (!mBoundService.isServiceAvailable()) {
            mBoundService.setupService();
            mBoundService.startService(BluetoothState.DEVICE_ANDROID);
            setup();
        }
    }

}*/




public void onDestroy() {
    super.onDestroy();
    bt.stopService();
}


public boolean onCreateOptionsMenu(Menu menu) {
    this.menu = menu;
    getMenuInflater().inflate(R.menu.menu_connection, menu);
    return true;
}


public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    if (id == R.id.menu_android_connect) {
        bt.setDeviceTarget(BluetoothState.DEVICE_ANDROID);
        /*
                        if(bt.getServiceState() ==
BluetoothState.STATE_CONNECTED)
                        bt.disconnect();*/
        Intent intent = new Intent(getApplicationContext(), DeviceList.class);
        startActivityForResult(intent, BluetoothState.REQUEST_CONNECT_DEVICE);
    } else if (id == R.id.menu_device_connect) {
        bt.setDeviceTarget(BluetoothState.DEVICE_OTHER);
                        /*
                        if(bt.getServiceState() ==
BluetoothState.STATE_CONNECTED)
                        bt.disconnect();*/
        Intent intent = new Intent(getApplicationContext(), DeviceList.class);
        startActivityForResult(intent, BluetoothState.REQUEST_CONNECT_DEVICE);
    } else if (id == R.id.menu_disconnect) {
        if (bt.getServiceState() == BluetoothState.STATE_CONNECTED)
            bt.disconnect();
    }


    else if (id == R.id.menu_reinitialize) {
        hrData.setText("");
        oximetryData.setText("");
    }
    return super.onOptionsItemSelected(item);
}
public void onStart() {
    super.onStart();
    if (!bt.isBluetoothEnabled()) {
        Intent intent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(intent, BluetoothState.REQUEST_ENABLE_BT);
    } else {
        if (!bt.isServiceAvailable()) {
            bt.setupService();
            bt.startService(BluetoothState.DEVICE_ANDROID);
            setup();
        }
    }


}
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == BluetoothState.REQUEST_CONNECT_DEVICE) {
        if (resultCode == Activity.RESULT_OK)
            bt.connect(data);
    } else if (requestCode == BluetoothState.REQUEST_ENABLE_BT) {
        if (resultCode == Activity.RESULT_OK) {
            bt.setupService();
            bt.startService(BluetoothState.DEVICE_ANDROID);
            setup();
        } else {
            Toast.makeText(getApplicationContext()
                    , "BluetoothActivity was not enabled."
                    , Toast.LENGTH_SHORT).show();
            finish();
        }
    }
}
public void setup() {


    test.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            {
                bt.send("PD", true);




            }
        }
    });
}





@Override
public void onStop() {
    super.onStop();


}
}
   package
   nsf.esarplab.scc
   health;


                              import android.app.Activity;
                              import android.app.ProgressDialog;
                              import android.bluetooth.BluetoothAdapter;
                              import android.content.ContentValues;
                              import android.content.Context;
                              import android.content.DialogInterface;
                              import android.content.Intent;
                              import android.content.SharedPreferences;
                              import android.database.sqlite.SQLiteDatabase;
                              import android.graphics.Color;
                              import android.graphics.drawable.GradientDrawable;
                              import android.os.Bundle;
                              import android.os.CountDownTimer;
                              import android.os.Environment;
                              import android.os.Handler;
                              import android.support.v7.app.ActionBar;
                              import android.support.v7.app.AlertDialog;
                              import android.support.v7.app.AppCompatActivity;
                              import android.telephony.SmsManager;
                              import android.util.Log;
                              import android.view.Gravity;
                              import android.view.LayoutInflater;
                              import android.view.Menu;
                              import android.view.MenuItem;
                              import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.PopupWindow;
import android.widget.TextView;
import android.widget.Toast;


import com.jjoe64.graphview.GraphView;
import com.jjoe64.graphview.GraphView.GraphViewData;
import com.jjoe64.graphview.GraphViewSeries;
import com.jjoe64.graphview.LineGraphView;


import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.lang.reflect.Method;
import java.text.DateFormat;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;


import dalvik.system.DexClassLoader;
import nsf.esarplab.bluetoothlibrary.BluetoothSPP;
import nsf.esarplab.bluetoothlibrary.BluetoothState;
import nsf.esarplab.bluetoothlibrary.DeviceList;


import static nsf.esarplab.scchealth.R.id.graph1;


public class FluActivity extends AppCompatActivity {
   BluetoothSPP bt;
   double ratingOfEOI = 0.0;
   Float severityRating;
   String s="";
   Intent mIntent;
   private final Handler mHandler = new Handler();
   private Runnable mTimer1;
   private int fileSeq=1;
   private TextView connectionRead;
   private TextView mNameText;
   private TextView mDateTime;
   private TextView mTemperature;
   private TextView mEoi;
   private Button connectScanner, test, dispResult;
   //private WifiManager wifiManager;
   private boolean diseaseKey = false;
   private boolean sensorKey = false;
   private boolean timeKey = false;
   private boolean connected=false;
```

```
    private int sensor = 1;
    private String tempReceived, eoiValue, sSeverity;
    private String currentDateTime = "";
    private TextView Vdatetime, gradient, Textv, Veoi, Vprompt;
    //String mealId = " ";
    private String eoiRating = "0";
    private String prompt = " ";
    private String sEOI="";
    private String sTemperature="";
    private Menu menu;
    private ImageView arrow1, arrow2, arrow3, arrow4, arrow5,
arrow6, arrow7, arrow8, arrow9, arrow10, arrow11;
    private LinearLayout mDisplay, graph;
    private ArrayList<String> arr_hex = new ArrayList<String>();
    private ArrayList<Short> arr_received = new ArrayList<Short>();
    private ProgressDialog progressDialog;
    private CountDownTimer Count;
    private PopupWindow pw;
    private GraphView graphView1;
    private GraphViewSeries exampleSeries1;
    private double sensorX = 0;
    private List<GraphViewData> seriesX;
    int dataCount = 1;




    //temp db finish


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);


        bt = new BluetoothSPP(this);
        //
((cBaseApplication)this.getApplicationContext()).myBlueComms.Bl
uetoothConnectionListener();


       /* if(bt.isBluetoothEnabled())
            Toast.makeText(this,"b
on",Toast.LENGTH_SHORT).show();
        else
            Toast.makeText(this,"b
off",Toast.LENGTH_SHORT).show();
*/
        setContentView(R.layout.activity_body_temp);


        // show action bar
        ActionBar myActionBar = getSupportActionBar();
        myActionBar.show();
        /*startService(new Intent(this, BluetoothSPP.class));
        bindService(mIntent, mConnection,
BIND_AUTO_CREATE);*/


        // Find all relevant views that we will need to read user input
from


        mNameText = (TextView) findViewById(R.id.display_name);
        mDateTime = (TextView) findViewById(R.id.datetime);
        mTemperature = (TextView) findViewById(R.id.display_bdt);
        mEoi = (TextView) findViewById(R.id.display_eoi);
        test = (Button) findViewById(R.id.test);
        dispResult = (Button) findViewById(R.id.displayResult);
        connectScanner = (Button) findViewById(R.id.cScanner);
        Vdatetime = (TextView) findViewById(R.id.datetime);
        Textv = (TextView) findViewById(R.id.display_bdt);
        Veoi = (TextView) findViewById(R.id.display_eoi);
        Vprompt = (TextView) findViewById(R.id.display_prompt);
        connectionRead = (TextView) findViewById(R.id.textStatus);
        mDisplay = (LinearLayout) findViewById(R.id.maindisplay);
        graph=(LinearLayout) findViewById(R.id.graph1);
        arrow1 = (ImageView) findViewById(R.id.arrow1);
        arrow2 = (ImageView) findViewById(R.id.arrow2);
        arrow3 = (ImageView) findViewById(R.id.arrow3);
        arrow4 = (ImageView) findViewById(R.id.arrow4);
        arrow5 = (ImageView) findViewById(R.id.arrow5);
        arrow6 = (ImageView) findViewById(R.id.arrow6);
        arrow7 = (ImageView) findViewById(R.id.arrow7);
        arrow8 = (ImageView) findViewById(R.id.arrow8);
        arrow9 = (ImageView) findViewById(R.id.arrow9);
        arrow10 = (ImageView) findViewById(R.id.arrow10);
        arrow11 = (ImageView) findViewById(R.id.arrow11);


        // Set active profile


        //Show active profile
        SharedPreferences prefs =
getSharedPreferences("logindetails",MODE_PRIVATE);
        String Uname =  prefs.getString("loginname","Default");
        mNameText.setText("\t\t"+Uname);


        s = mNameText.getText().toString().trim();


        //show graph


        seriesX = new ArrayList<GraphViewData>();
        // init example series data
        exampleSeries1 = new GraphViewSeries(new
GraphViewData[] {});


            graphView1 = new LineGraphView(
                    this // context
                    , "Real time plot" // heading
            );


        graphView1.addSeries(exampleSeries1); // data
        LinearLayout layout = (LinearLayout) findViewById(graph1);
        layout.addView(graphView1);


        /*
        //reading text from file
        try {
            FileInputStream fileIn = openFileInput("mytextfile.txt");
            InputStreamReader InputRead = new
InputStreamReader(fileIn);
            char[] inputBuffer = new char[READ_BLOCK_SIZE];
            *//*String s="";*//*
            int charRead;

            while ((charRead = InputRead.read(inputBuffer)) > 0) {
                // char to string conversion
                String readstring = String.copyValueOf(inputBuffer, 0,
charRead);
                s += readstring;
            }
            InputRead.close();
            *//*mNameText.setText(s);*//*
            *//*Toast.makeText(getBaseContext(),
s,Toast.LENGTH_SHORT).show();*//*

        } catch (Exception e) {
            e.printStackTrace();
        }
        mNameText.setText(s);*/



        // hide main display


        mDisplay.setVisibility(View.INVISIBLE);
        graph.setVisibility(View.INVISIBLE);
        arrow1.setVisibility(View.INVISIBLE);
        arrow2.setVisibility(View.INVISIBLE);
        arrow3.setVisibility(View.INVISIBLE);
        arrow4.setVisibility(View.INVISIBLE);
        arrow5.setVisibility(View.INVISIBLE);
        arrow6.setVisibility(View.INVISIBLE);
        arrow7.setVisibility(View.INVISIBLE);
        arrow8.setVisibility(View.INVISIBLE);
        arrow9.setVisibility(View.INVISIBLE);
        arrow10.setVisibility(View.INVISIBLE);
        arrow11.setVisibility(View.INVISIBLE);




// Find the View that shows the save button
        /*Button save = (Button) findViewById(R.id.save);

        // Set a click listener on that View
        save.setOnClickListener(new View.OnClickListener() {
            // The code in this method will be executed when the
numbers category is clicked on.
            @Override
            public void onClick(View view) {
                // Save record to database
                insertTemp();
                // Exit activity
                finish();

            }
        });*/




        // color line gradient




        gradient = (TextView) findViewById(R.id.active_gradient);




        int[] colors = {Color.parseColor("#008000"),
Color.parseColor("#FFFF00"), Color.parseColor("#FFA500"),
Color.parseColor("#ff0000"), Color.parseColor("#800000")};
        GradientDrawable gd = new GradientDrawable(
                GradientDrawable.Orientation.LEFT_RIGHT, colors);
        gradient.setBackground(gd);


        // pop up window for info and instruction


        final TextView btnOpenPopup = (TextView)
findViewById(R.id.info);
        btnOpenPopup.setOnClickListener(new
Button.OnClickListener() {


            @Override
```

```
          public void onClick(View arg0) {
            LayoutInflater layoutInflater
                = (LayoutInflater) getBaseContext()

.getSystemService(LAYOUT_INFLATER_SERVICE);
            View popupView =
layoutInflater.inflate(R.layout.info_bodytemp, null);
            final PopupWindow popupWindow = new PopupWindow(
                popupView,
                ViewGroup.LayoutParams.WRAP_CONTENT,
                ViewGroup.LayoutParams.WRAP_CONTENT);


            Button btnDismiss = (Button)
popupView.findViewById(R.id.dismiss);
            btnDismiss.setOnClickListener(new
Button.OnClickListener() {


              @Override
              public void onClick(View v) {
                // TODO Auto-generated method stub
                popupWindow.dismiss();
              }
            });


            popupWindow.showAsDropDown(btnOpenPopup, 50, -
30);


          }
        });


        final TextView btnOpenInstruction = (TextView)
findViewById(R.id.inst);
        btnOpenInstruction.setOnClickListener(new
Button.OnClickListener() {


          @Override
          public void onClick(View arg0) {
            LayoutInflater layoutInflater
                = (LayoutInflater) getBaseContext()

.getSystemService(LAYOUT_INFLATER_SERVICE);
            View popupView =
layoutInflater.inflate(R.layout.inst_bodytemp, null);
            final PopupWindow popupWindow = new PopupWindow(
                popupView,
                ViewGroup.LayoutParams.WRAP_CONTENT,
                ViewGroup.LayoutParams.WRAP_CONTENT);
            Button btnDismiss = (Button)
popupView.findViewById(R.id.dismiss);
            btnDismiss.setOnClickListener(new
Button.OnClickListener() {


              @Override
              public void onClick(View v) {
                // TODO Auto-generated method stub
                popupWindow.dismiss();
              }
            });


            popupWindow.showAsDropDown(btnOpenPopup, 50, -
30);


        }
      });


      // connect scanner by bluetooth


      connectScanner.setOnClickListener(new
View.OnClickListener() {
          @Override
          public void onClick(View view) {
            bt.setDeviceTarget(BluetoothState.DEVICE_OTHER);
          /*
                              if(bt.getServiceState() ==
BluetoothState.STATE_CONNECTED)
                              bt.disconnect();*/
            Intent intent = new Intent(getApplicationContext(),
DeviceList.class);
            startActivityForResult(intent,
BluetoothState.REQUEST_CONNECT_DEVICE);


        }
      });


      dispResult.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
          tempAlgorithm(v);


        }
      });


      //EOI color ranking
```

```
        //LinearLayout mainDisplay=(LinearLayout)
findViewById(R.id.maindisplay);



        //mainDisplay.setVisibility(View.INVISIBLE);



        // set the bluetooth connection


        bt.setOnDataReceivedListener(new
BluetoothSPP.OnDataReceivedListener() {
            public void onDataReceived(byte[] data, String message) {
                short val = 0;
                String readAscii = new String(data);
                //Log.i("Str@activity", readAscii);
                //textReceived.append(message + "\n");
                if (timeKey) {
                    //receive data


                    arr_hex.add(message);
                    Log.i("size_arr_hex",""+arr_hex.size());




                    if (arr_hex.size() == 2) {
                        String catHex = arr_hex.get(0) + arr_hex.get(1);
                        /*int b0 = (arr_hex.get(0) & 255); // converts to
unsigned
                        int b1 = (arr_hex.get(1) & 255); // converts to
unsigned
                        int val = b0 << 8 | b1;*/
                        Log.i("val1@final", catHex);
                        val = (short) (Integer.parseInt(catHex, 16));
                        Log.i("val2@final", String.valueOf(val));
                        arr_received.add(val);
                        //Textv.append(Integer.toString(val) + "\n");
                        seriesX.add(new GraphViewData(dataCount, val));
                        dataCount++;
                        if (arr_received.size() > 600) {
                            seriesX.remove(0);
                            graphView1.setViewPort(dataCount - 600, 600);


                        }
                        try {
                            writeToCsv(Integer.toString(val));
                        } catch (IOException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                    }
                    arr_hex.clear();
                    Log.i("sizearr_received", "" + arr_received.size());
                  }



                } else {
                  if (readAscii.equals("OS")) {


                    bt.send("TP", true);
                    diseaseKey = true;


                  } else {
                    if (readAscii.equals("TP") && diseaseKey) {
                      bt.send("00020", true);
                      sensorKey = true;
                    } else {
                      if (readAscii.equals("00020") && diseaseKey &&
sensorKey) {
                        bt.send("OK", true);
                        timeKey = true;


                      } else {
                        mDisplay.setVisibility(View.VISIBLE);
                        arrow1.setVisibility(View.INVISIBLE);
                        arrow2.setVisibility(View.INVISIBLE);
                        arrow3.setVisibility(View.INVISIBLE);
                        arrow4.setVisibility(View.INVISIBLE);
                        arrow5.setVisibility(View.INVISIBLE);
                        arrow6.setVisibility(View.INVISIBLE);
                        arrow7.setVisibility(View.INVISIBLE);
                        arrow8.setVisibility(View.INVISIBLE);
                        arrow9.setVisibility(View.INVISIBLE);
                        arrow10.setVisibility(View.INVISIBLE);
                        arrow11.setVisibility(View.INVISIBLE);
                        Toast.makeText(getApplicationContext(),
"Failed Handshake", Toast.LENGTH_LONG).show();
                        Count.cancel();
                        progressDialog.dismiss();


                      }
                    }
                  }
                }
              }


            });
```

```
      bt.setBluetoothConnectionListener(new
BluetoothSPP.BluetoothConnectionListener() {
          public void onDeviceDisconnected() {
             connectionRead.setText("Status : Not connect");
             connected=false;
             menu.clear();
             getMenuInflater().inflate(R.menu.menu_connection,
menu);
          }


          public void onDeviceConnectionFailed() {
             connectionRead.setText("Status : Connection failed");
             AlertDialog.Builder builder = new
AlertDialog.Builder(FluActivity.this);
             builder.setTitle("Connection Error");
             builder.setMessage("Retry to connect");


             // add the buttons
             builder.setPositiveButton("Retry", new
DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                   // do something ...

bt.setDeviceTarget(BluetoothState.DEVICE_OTHER);
                   Intent intent = new Intent(getApplicationContext(),
DeviceList.class);
                   startActivityForResult(intent,
BluetoothState.REQUEST_CONNECT_DEVICE);
                   dialog.dismiss();
                }
             });
             builder.setNegativeButton("Cancel", null);


             // create and show the alert dialog
             AlertDialog dialog = builder.create();
             dialog.show();
          }


          public void onDeviceConnected(String name, String address)
{
             connectionRead.setText("Status : Connected to " + name);
             connected=true;
             menu.clear();
             getMenuInflater().inflate(R.menu.menu_disconnection,
menu);
          }
      });
```

```
   }


   /**
    * save new entry  into database.
    */
   private void insertTemp() {
      // Read from input fields
      // Use trim to eliminate leading or trailing white space
      String nameString = mNameText.getText().toString().trim();
      String dateString = mDateTime.getText().toString().trim();
      String valueString = mTemperature.getText().toString().trim();
      String eoiString = eoiValue;




      // Create database helper
      TempDbHelper mDbHelper = new TempDbHelper(this);


      // Gets the database in write mode
      SQLiteDatabase db = mDbHelper.getWritableDatabase();


      // Create a ContentValues object where column names are the
keys,
      // and attributes from the editor are the values.
      ContentValues values = new ContentValues();

values.put(TempContract.TempEntry.COLUMN_PATIENT_NAME
, nameString);

values.put(TempContract.TempEntry.COLUMN_DATE_TIME,
dateString);

values.put(TempContract.TempEntry.COLUMN_TEMP_VALUE,
sTemperature);

values.put(TempContract.TempEntry.COLUMN_EOI_RATING,
sEOI);


      // Insert a new row  in the database, returning the ID of that new
row.
      long newRowId =
db.insert(TempContract.TempEntry.TABLE_NAME, null, values);


      // Show a toast message depending on whether or not the
insertion was successful
      if (newRowId == -1) {
         // If the row ID is -1, then there was an error with insertion.
         Toast.makeText(this, "Error with saving",
Toast.LENGTH_SHORT).show();
      } else {
          // Otherwise, the insertion was successful and we can display
a toast with the row ID.
          Toast.makeText(this, "saved with row id: " + newRowId,
Toast.LENGTH_SHORT).show();
        }
    }




    public void onDestroy() {
        super.onDestroy();
        bt.stopService();
    }




    // menu options


    public boolean onCreateOptionsMenu(Menu menu) {
        this.menu = menu;
        getMenuInflater().inflate(R.menu.menu_connection, menu);
        return true;
    }


    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {


            case R.id.menu_android_connect:
                bt.setDeviceTarget(BluetoothState.DEVICE_ANDROID);
            /*
            if(bt.getServiceState() ==
BluetoothState.STATE_CONNECTED)
                                bt.disconnect();*/
                Intent intent = new Intent(getApplicationContext(),
DeviceList.class);
                startActivityForResult(intent,
BluetoothState.REQUEST_CONNECT_DEVICE);
                return true;




            case R.id.menu_device_connect:
                bt.setDeviceTarget(BluetoothState.DEVICE_OTHER);
                                /*
                                if(bt.getServiceState() ==
BluetoothState.STATE_CONNECTED)
                                bt.disconnect();*/
                Intent intent2 = new Intent(getApplicationContext(),
DeviceList.class);
```

```
            startActivityForResult(intent2,
BluetoothState.REQUEST_CONNECT_DEVICE);
            return true;


         case R.id.menu_disconnect:
            if (bt.getServiceState() ==
BluetoothState.STATE_CONNECTED)
               bt.disconnect();




            return true;




         case R.id.menu_reinitialize:
            Textv.setText("");
            Vdatetime.setText("");
            Vprompt.setText("");
            Veoi.setText("");
            graph.setVisibility(View.INVISIBLE);
            arr_received.clear();
            seriesX.clear();
            dataCount = 1;
            mDisplay.setVisibility(View.INVISIBLE);
            diseaseKey = false;
            sensorKey = false;
            timeKey = false;
            return true;


         case R.id.action_save:
            // Save record to database
            insertTemp();
            // Exit activity
            //finish();
            return true;
         // Respond to a click on the "Share to SCC" menu option
         case R.id.action_share:




            if(s.matches("")) {
               Toast.makeText(getApplicationContext(), "Create
Profile First ", Toast.LENGTH_LONG).show();


            }else{


               // Go to cloud activity
```

```
                Intent shareIntent = new Intent(FluActivity.this,
CloudActivity.class);
                //Bundle extras = new Bundle();
                shareIntent.putExtra("DT", "BT");
                shareIntent.putExtra("profile", s);
                shareIntent.putExtra("EOI", eoiValue);
                shareIntent.putExtra("Time", currentDateTime);
                shareIntent.putExtra("Algorithm", "BT1");
                startActivity(shareIntent);
                return true;


            }
          case R.id.action_sms:
             String messageToSend = "EOI:" + eoiValue;
             String number = "9015157371";


             SmsManager.getDefault().sendTextMessage(number, null,
messageToSend, null, null);
             //finish();
             return true;
          case R.id.action_history:


             // Create a new intent to open the {@link Temperature
History}
             Intent temperatureHistoryIntent = new
Intent(FluActivity.this, Temp_HistoryActivity.class);


             // Start the new activity
             startActivity(temperatureHistoryIntent);
             //finish();
             return true;


          case R.id.action_algorithm:
             Intent algIntent = new Intent(FluActivity.this,
FluMethods.class);
             startActivity(algIntent);
             return true;
       }
       return super.onOptionsItemSelected(item);
    }


    public void onStart() {
       super.onStart();
       if (!bt.isBluetoothEnabled()) {
          Intent intent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
          startActivityForResult(intent,
BluetoothState.REQUEST_ENABLE_BT);
       } else {
          if (!bt.isServiceAvailable()) {
                bt.setupService();
                bt.startService(BluetoothState.DEVICE_ANDROID);
                setup();
            }
        }


    }


    public void onActivityResult(int requestCode, int resultCode,
Intent data) {
        if (requestCode ==
BluetoothState.REQUEST_CONNECT_DEVICE) {
            if (resultCode == Activity.RESULT_OK)
                bt.connect(data);
        } else if (requestCode ==
BluetoothState.REQUEST_ENABLE_BT) {
            if (resultCode == Activity.RESULT_OK) {
                bt.setupService();
                bt.startService(BluetoothState.DEVICE_ANDROID);
                setup();
            } else {
                Toast.makeText(getApplicationContext()
                        , "BluetoothActivity was not enabled."
                        , Toast.LENGTH_SHORT).show();
                finish();
            }
        }
    }
    @Override
    protected void onResume() {
        super.onResume();
        mTimer1 = new Runnable() {
            @Override
            public void run() {
                GraphViewData[] gvd = new
GraphViewData[seriesX.size()];
                seriesX.toArray(gvd);
                exampleSeries1.resetData(gvd);
                mHandler.post(this); //, 100);
            }
        };
        mHandler.postDelayed(mTimer1, 100);


    }


    public void setup() {


        test.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                {
                    if (connected) {
```

```
                    bt.send("OS", true);
                    graph.setVisibility(View.VISIBLE);
                    // progress indicator
                    progressDialog = new
ProgressDialog(FluActivity.this,
                            R.style.AppTheme_Dark_Dialog);
                    progressDialog.setIndeterminate(true);
                    progressDialog.setMessage("Collecting data...");


                    Count=new CountDownTimer(500, 100) {


                        public void onTick(long millisecondsUntilDone) {


                            progressDialog.show();
                        }


                        @Override
                        public void onFinish() {
                            Log.i("Done", "Count Down Timer Finished");
                            progressDialog.dismiss();


                        }
                    }.start();


                } else {
                    Toast.makeText(getApplicationContext(), "Get
Connected First", Toast.LENGTH_SHORT).show();
                }
            }
        }
    });
}




@Override
public void onStop() {
    super.onStop();




}




public void tempAlgorithm(View v) {
    ArrayList<Short> arr_trans = new ArrayList<Short>();
    ArrayList<Short> arr_processed1 = new ArrayList<Short>();
    ArrayList<Short> arr_processed2 = new ArrayList<Short>();
    float sum = 0.0f;
    float sum1 = 0.0f;
    float sum2 = 0.0f;
    float avgValue = 0.0f;
    float avgValue1 = 0.0f;
    float avgValue2 = 0.0f;
    float resultVoltage=0.0f;
    double temperature=0.0f;


    // make main display visible and hide arrows
    mDisplay.setVisibility(View.VISIBLE);
    arrow1.setVisibility(View.INVISIBLE);
    arrow2.setVisibility(View.INVISIBLE);
    arrow3.setVisibility(View.INVISIBLE);
    arrow4.setVisibility(View.INVISIBLE);
    arrow5.setVisibility(View.INVISIBLE);
    arrow6.setVisibility(View.INVISIBLE);
    arrow7.setVisibility(View.INVISIBLE);
    arrow8.setVisibility(View.INVISIBLE);
    arrow9.setVisibility(View.INVISIBLE);
    arrow10.setVisibility(View.INVISIBLE);
    arrow11.setVisibility(View.INVISIBLE);


    // display when there is no data
    if (arr_received.size() == 0) {
        mDisplay.setVisibility(View.VISIBLE);
        Textv.setText("No Data");
        // Veoi.setText("No Data");
    } else {


        // get date and time


        currentDateTime =
DateFormat.getDateTimeInstance().format(new Date());
        // initialize the screen
        Vdatetime.setText("");
        Textv.setText("");
        //Vprompt.setText("");
        //Veoi.setText("");


        // temperature processing begin


       /* for (int i = 0; i < arr_received.size(); i++) {
            if ((arr_received.get(i)>0)&&(arr_received.get(i)<9000)) {
                arr_trans.add(arr_received.get(i));
                Log.i("transferrred", "" + arr_received.get(i));

            }
        }*/




        short value=arr_received.get(0);
        for (int i=0;i<arr_received.size();i++)
        {
            short currentValue=arr_received.get(i);
            value+=((currentValue - value)/10);
            arr_processed1.add(i,value);
            try {
                writeToCsv(Integer.toString(value));
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }




        // *****Feature 2
*******************************************************
***************
        // step-1-find maxima


        int max =arr_processed1.get(0);
        for (int l=0;l<arr_processed1.size(); l++){
            if(arr_processed1.get(l)> max){
                max = arr_processed1.get(l);
            }
        }
        Log.i("feature21", "" + max);
        // step-2-find maxima index


        int indexOfMaxima=0;


        for (int m=0; m<arr_processed1.size(); m++)


        {
            if (max==arr_processed1.get(m)){
                indexOfMaxima=m;
                break;
            }
        }


        // step-3-find maxima level
        float sumMaxima=0;
        for (int n=indexOfMaxima-10; n<indexOfMaxima+10; n++)


        {
          sumMaxima+=arr_processed1.get(n);
        }
        float avgMaxima=sumMaxima/20;
        Log.i("feature22", "" + avgMaxima);


        // ****feature
3*****************************************************
*****************


        // step-1-find index of delay


        for (int q = 99; q < arr_processed1.size(); q++) {


          arr_trans.add(arr_processed1.get(q));


        }


        // step-2-find index of delay
        int indexOfDelay=0;


        for (int p=0; p<arr_trans.size(); p++)


        {
          if (((arr_trans.get(p))-2450)<5){
            indexOfDelay=p;
            break;
          }
        }
        Log.i("feature3", "" + indexOfDelay);


        // *****Feature 1
*****************************************************
*************


        // step-1-find minima


        int min =arr_trans.get(0);
        for (int i=0;i<arr_trans.size(); i++){
            if(arr_trans.get(i)< min){
                min = arr_trans.get(i);
            }
        }
        //System.out.println(min);


        // step-2-find minima index


        int indexOfMinima=0;


        for (int j=0; j<arr_trans.size(); j++)


        {
            if (min==arr_trans.get(j)){
                indexOfMinima=j;
                break;
            }
        }
        Log.i("feature11", "" + min);


        // step-3-find minima level


        float sumMinima=0;
        for (int k=indexOfMinima; k<indexOfMinima+10; k++)


        {
            sumMinima+=arr_trans.get(k);
        }
        float avgMinima=sumMinima/10;
        Log.i("feature12", "" + avgMinima);


        // ******Feature 4
*******************************************************
*************


        for (int s = 0; s < arr_trans.size(); s++) {
            sum += arr_trans.get(s);
        }
        avgValue = sum / arr_trans.size();


        Log.i("feature4", "" + avgValue);


        // ********** Multivariate regression
*************************
        // equation for temperature
        temperature= 228.6-0.04243*avgMaxima-
0.21267*indexOfDelay;


        double temp2=230.0-0.00142*avgMinima-
0.04203*avgMaxima-0.21037*indexOfDelay;
        Log.i("temp", "" + temp2);
        Log.i("sizer", "" + arr_received.size());
        Log.i("sizet", "" + arr_trans.size());
        Log.i("sizep", "" + arr_processed1.size());




        try {
            sTemperature=String.valueOf(new
DecimalFormat("###.##").format(temperature));
            ratingOfEOI = (temperature - 97) / 10;
            if(ratingOfEOI<0){
                ratingOfEOI=0;
            }else if(ratingOfEOI>1){
                ratingOfEOI=1;
            }
            sEOI = new
DecimalFormat("##.##").format(ratingOfEOI);
            sSeverity = new DecimalFormat("##.##").format(100 *
ratingOfEOI);


            if (temperature<= 97.5) {
                prompt = "Normal Temperature";
                arrow1.setVisibility(View.VISIBLE);
                sEOI="0.0";
                sSeverity="0.0";
            } else if (temperature <= 98.5) {
                prompt = "Normal Temperature";
                arrow2.setVisibility(View.VISIBLE);
            } else if (temperature <= 99.5) {
                prompt = "Normal Temperature";
                arrow3.setVisibility(View.VISIBLE);
            } else if (temperature <= 100.5) {
                prompt = "Normal Temperature";
                arrow4.setVisibility(View.VISIBLE);
            } else if (temperature <= 101.5) {
                prompt = "Low Fever,\nconsider consulting your
doctor";
                arrow5.setVisibility(View.VISIBLE);
            } else if (temperature <= 102.5) {
                prompt = "Medium Fever,\nConsult your doctor";
                arrow6.setVisibility(View.VISIBLE);
            } else if (temperature <= 103.5) {
                prompt = "High Fever,\nConsult your doctor";
                arrow7.setVisibility(View.VISIBLE);
            } else if (temperature <= 104.5) {
                prompt = "High Fever,\nConsult your doctor";
                arrow8.setVisibility(View.VISIBLE);
```

```
              } else if (temperature <= 105.5) {
                prompt = "Very High Fever,\nConsult your doctor
immediately";
                arrow9.setVisibility(View.VISIBLE);
              } else if (temperature <= 106.5) {
                prompt = "Very High Fever,\nConsult your doctor
immediately";
                arrow10.setVisibility(View.VISIBLE);
              } else if (temperature >= 106.5) {
                prompt = "Extremely High Fever,\nConsult your doctor
immediately";
                arrow11.setVisibility(View.VISIBLE);
              }


            } catch (NumberFormatException e) {


              prompt = "Invalid Data";
              gradient.setVisibility(View.INVISIBLE);
            }





//------ displaying result


            Vdatetime.setText(currentDateTime);


            //Textv.append(sTemperature+"°F");
            Textv.append(String.format("%.1f",temperature)+"°F");


            Vprompt.append(prompt );
            //Veoi.append("fluSeverity(100) = " + result);


            Veoi.append( sSeverity);
          }


          // end temperature processing
          arr_received.clear();
          Log.i("sizearr_received", "" + arr_received.size());
          arr_trans.clear();
          arr_processed1.clear();
          arr_processed2.clear();
          fileSeq++;
          timeKey = false;
          diseaseKey = false;
          sensorKey = false;
        }
```

```
    private void initiatePopupWindow(View v) {
        try {
            //We need to get the instance of the LayoutInflater, use the
context of this activity
            LayoutInflater inflater = (LayoutInflater) FluActivity.this

.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            //Inflate the view from a predefined XML layout
            View layout = inflater.inflate(R.layout.flu_symp,
                    (ViewGroup) findViewById(R.id.popup_element));
            // create a 300px width and 470px height PopupWindow
            pw = new PopupWindow(layout, 300, 470, true);
            // display the popup in the center
            pw.showAtLocation(v, Gravity.CENTER, 0, 0);




        } catch (Exception e) {
            e.printStackTrace();
        }
    }




    //write to csv file
    public void writeToCsv(String x) throws IOException {


        Calendar c = Calendar.getInstance();
        File folder = new
File(Environment.getExternalStorageDirectory() + "/project");
        boolean success = true;
        if (!folder.exists()) {
            success = folder.mkdir();
        }
        if (success) {
            // Do something on success
            String fileName = "flu" + String.valueOf(fileSeq) + ".csv";
            String csv = "/storage/emulated/0/project/"+fileName;
            FileWriter file_writer = new FileWriter(csv, true);
            String s = c.get(Calendar.YEAR) + "," +
(c.get(Calendar.MONTH) + 1) + "," + c.get(Calendar.DATE) + "," +
c.get(Calendar.HOUR) + "," + c.get(Calendar.MINUTE) + "," +
c.get(Calendar.SECOND) + "," + c.get(Calendar.MILLISECOND) +
"," + x + "\n";


            file_writer.append(s);
            file_writer.close();
        }
    }


    public String dexcallFluSeverity(Integer temp) {
        try {
            final String libPath =
Environment.getExternalStoragePublicDirectory(Environment.DIRE
CTORY_DOWNLOADS ) + "/fludex.dex";//path to DEX file to
load
            final File tmpDir = getDir("dex", 0);//temp directory
optimized dex files should be written
            final DexClassLoader classloader = new
DexClassLoader(libPath, tmpDir.getAbsolutePath(), null,
this.getClass().getClassLoader());//create DexClassLoader object
            final Class<Object> classToLoad = (Class<Object>)
classloader.loadClass("com.example.eoiValue");//load class with
class name - eoiValue
            final Object myInstance  =
classToLoad.newInstance();//create a instance of the class loaded
above
            final Method doSomething =
classToLoad.getMethod("fluSeverity", Integer.class);//get method of
the class loaded above


            String result  = (String) doSomething.invoke(myInstance,
temp);//finally, invoke the method of the instance, while passing
parameter value
            return result;//return the method invocation result
        } catch (Exception e) {
            e.printStackTrace();
        }


        return null;
    }


}
ackage nsf.esarplab.scchealth;


import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
```

```
public class BluetoothActivity extends
AppCompatActivity implements OnClickListener {



    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.bluetooth);


        // show action bar
        ActionBar myActionBar = getSupportActionBar();
        myActionBar.show();


        /*Button btnSimple = (Button)
findViewById(R.id.btnSimple);
        btnSimple.setOnClickListener(this);

        Button btnListener = (Button)
findViewById(R.id.btnListener);
        btnListener.setOnClickListener(this);

        Button btnAutoConnect = (Button)
findViewById(R.id.btnAutoConnect);
        btnAutoConnect.setOnClickListener(this);*/


        Button btnDeviceList = (Button)
findViewById(R.id.btnDeviceList);
        btnDeviceList.setOnClickListener(this);


        Button btnTerminal = (Button)
findViewById(R.id.btnTerminal);
        btnTerminal.setOnClickListener(this);


        Button btnPair = (Button)
findViewById(R.id.btnPair);
        btnPair.setOnClickListener(this);


    }


    public void onClick(View v) {
        int id = v.getId();
        Intent intent = null;
        switch (id) {
            /*case R.id.btnSimple:
                intent = new Intent(getApplicationContext(),
SimpleActivity.class);
                startActivity(intent);
                break;
            case R.id.btnListener:
                intent = new Intent(getApplicationContext(),
ListenerActivity.class);
                startActivity(intent);
                break;
            case R.id.btnAutoConnect:
                intent = new Intent(getApplicationContext(),
AutoConnectActivity.class);
                startActivity(intent);
                break;*/
            case R.id.btnDeviceList:
                intent = new Intent(getApplicationContext(),
DeviceListActivity.class);
                startActivity(intent);
                break;
            case R.id.btnTerminal:
                intent = new Intent(getApplicationContext(),
TerminalActivity.class);
                startActivity(intent);
                break;
            case R.id.btnPair:
                intent = new Intent(getApplicationContext(),
BluetoothPair.class);
                startActivity(intent);
                break;
        }
    }
}
package
nsf.esarplab.scchealth;


import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothClass;
import android.bluetooth.BluetoothDevice;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.Toast;


import java.lang.reflect.Method;
import java.util.ArrayList;
import java.util.Set;
```

```
public class BluetoothPair extends AppCompatActivity {
    ListView listViewPaired;
    ListView listViewDetected;
    ArrayList<String> arrayListpaired;
    Button buttonSearch,buttonOn,buttonDesc,buttonOff;
    ArrayAdapter<String> adapter,detectedAdapter;
    static HandleSeacrh handleSeacrh;
    BluetoothDevice bdDevice;
    BluetoothClass bdClass;
    ArrayList<BluetoothDevice> arrayListPairedBluetoothDevices;
    private ButtonClicked clicked;
    ListItemClickedonPaired listItemClickedonPaired;
    BluetoothAdapter bluetoothAdapter = null;
    ArrayList<BluetoothDevice> arrayListBluetoothDevices = null;
    ListItemClicked listItemClicked;


    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_bluetooth_pair);
        listViewDetected = (ListView) findViewById(R.id.listViewDetected);
        listViewPaired = (ListView) findViewById(R.id.listViewPaired);
        buttonSearch = (Button) findViewById(R.id.buttonSearch);
        buttonOn = (Button) findViewById(R.id.buttonOn);
        buttonDesc = (Button) findViewById(R.id.buttonDesc);
        buttonOff = (Button) findViewById(R.id.buttonOff);
        arrayListpaired = new ArrayList<String>();
        bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        clicked = new ButtonClicked();
        handleSeacrh = new HandleSeacrh();
        arrayListPairedBluetoothDevices = new ArrayList<BluetoothDevice>();
        /*
         * the above declaration is just for getting the paired bluetooth devices;
         * this helps in the removing the bond between paired devices.
         */
        listItemClickedonPaired = new ListItemClickedonPaired();
        arrayListBluetoothDevices = new ArrayList<BluetoothDevice>();
        adapter= new ArrayAdapter<String>(BluetoothPair.this,
android.R.layout.simple_list_item_1, arrayListpaired);
        detectedAdapter = new ArrayAdapter<String>(BluetoothPair.this,
android.R.layout.simple_list_item_single_choice);
        listViewDetected.setAdapter(detectedAdapter);
        listItemClicked = new ListItemClicked();
        detectedAdapter.notifyDataSetChanged();
        listViewPaired.setAdapter(adapter);
    }


    @Override
    protected void onStart() {
        // TODO Auto-generated method stub
        super.onStart();
        getPairedDevices();
        buttonOn.setOnClickListener(clicked);
        buttonSearch.setOnClickListener(clicked);
        buttonDesc.setOnClickListener(clicked);
        buttonOff.setOnClickListener(clicked);
        listViewDetected.setOnItemClickListener(listItemClicked);
        listViewPaired.setOnItemClickListener(listItemClickedonPaired);
    }
    private void getPairedDevices() {
        Set<BluetoothDevice> pairedDevice =
bluetoothAdapter.getBondedDevices();
        if(pairedDevice.size()>0)
        {
            for(BluetoothDevice device : pairedDevice)
            {
                arrayListpaired.add(device.getName()+"\n"+device.getAddress());
                arrayListPairedBluetoothDevices.add(device);
            }
        }
        adapter.notifyDataSetChanged();
    }
    class ListItemClicked implements AdapterView.OnItemClickListener
    {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {
            // TODO Auto-generated method stub
            bdDevice = arrayListBluetoothDevices.get(position);
            //bdClass = arrayListBluetoothDevices.get(position);
            Log.i("Log", "The dvice : "+bdDevice.toString());
            /*
             * here below we can do pairing without calling the callthread(), we can
directly call the
             * connect(). but for the safer side we must usethe threading object.
             */
            //callThread();
            //connect(bdDevice);
            Boolean isBonded = false;
            try {
                isBonded = createBond(bdDevice);
                if(isBonded)
                {

//arrayListpaired.add(bdDevice.getName()+"\n"+bdDevice.getAddress());
                    //adapter.notifyDataSetChanged();
                    getPairedDevices();
                    adapter.notifyDataSetChanged();
                }
            } catch (Exception e) {
                e.printStackTrace();
            }//connect(bdDevice);
            Log.i("Log", "The bond is created: "+isBonded);
        }
    }
    class ListItemClickedonPaired implements
AdapterView.OnItemClickListener
    {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int
position,long id) {
            bdDevice = arrayListPairedBluetoothDevices.get(position);
            try {
                Boolean removeBonding = removeBond(bdDevice);
                if(removeBonding)
                {
                    arrayListpaired.remove(position);
                    adapter.notifyDataSetChanged();
                }




                Log.i("Log", "Removed"+removeBonding);
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
    /*private void callThread() {
        new Thread(){
            public void run() {
                Boolean isBonded = false;
                try {
                    isBonded = createBond(bdDevice);
                    if(isBonded)
                    {

arrayListpaired.add(bdDevice.getName()+"\n"+bdDevice.getAddress());
                        adapter.notifyDataSetChanged();
                    }
                } catch (Exception e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }//connect(bdDevice);
                Log.i("Log", "The bond is created: "+isBonded);
            }
        }.start();
    }*/
    private Boolean connect(BluetoothDevice bdDevice) {
        Boolean bool = false;
        try {
            Log.i("Log", "service method is called ");
            Class cl = Class.forName("android.bluetooth.BluetoothDevice");
            Class[] par = {};
            Method method = cl.getMethod("createBond", par);
            Object[] args = {};
            bool = (Boolean) method.invoke(bdDevice);//, args);// this invoke creates
the detected devices paired.
            //Log.i("Log", "This is: "+bool.booleanValue());
            //Log.i("Log", "devicesss: "+bdDevice.getName());
        } catch (Exception e) {
            Log.i("Log", "Inside catch of serviceFromDevice Method");
            e.printStackTrace();
        }
        return bool.booleanValue();
    };




    public boolean removeBond(BluetoothDevice btDevice)
            throws Exception
    {
        Class btClass = Class.forName("android.bluetooth.BluetoothDevice");
        Method removeBondMethod = btClass.getMethod("removeBond");
        Boolean returnValue = (Boolean) removeBondMethod.invoke(btDevice);
        return returnValue.booleanValue();
    }




    public boolean createBond(BluetoothDevice btDevice)
            throws Exception
    {
        Class class1 = Class.forName("android.bluetooth.BluetoothDevice");
        Method createBondMethod = class1.getMethod("createBond");
        Boolean returnValue = (Boolean) createBondMethod.invoke(btDevice);
        return returnValue.booleanValue();
    }




    class ButtonClicked implements View.OnClickListener
    {
        @Override
        public void onClick(View view) {
            switch (view.getId()) {
                case R.id.buttonOn:
                    onBluetooth();
                    break;
                case R.id.buttonSearch:
                    arrayListBluetoothDevices.clear();
                    startSearching();
                    break;
                case R.id.buttonDesc:
                    makeDiscoverable();
                    break;
                case R.id.buttonOff:
                    offBluetooth();
                    break;
                default:
                    break;
            }
        }
    }
    private BroadcastReceiver myReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            Message msg = Message.obtain();
            String action = intent.getAction();
            if(BluetoothDevice.ACTION_FOUND.equals(action)){
                Toast.makeText(context, "ACTION_FOUND",
Toast.LENGTH_SHORT).show();


                BluetoothDevice device =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
                try
                {
                    //device.getClass().getMethod("setPairingConfirmation",
boolean.class).invoke(device, true);
                    //device.getClass().getMethod("cancelPairingUserInput",
boolean.class).invoke(device);
                }
                catch (Exception e) {
                    Log.i("Log", "Inside the exception: ");
                    e.printStackTrace();
                }


                if(arrayListBluetoothDevices.size()<1) // this checks if the size of
bluetooth device is 0,then add the
                {                                           // device to the arraylist.
                    detectedAdapter.add(device.getName()+"\n"+device.getAddress());
                    arrayListBluetoothDevices.add(device);
                    detectedAdapter.notifyDataSetChanged();
                }
                else
                {
                    boolean flag = true;    // flag to indicate that particular device is
already in the arlist or not
                    for(int i = 0; i<arrayListBluetoothDevices.size();i++)
                    {

if(device.getAddress().equals(arrayListBluetoothDevices.get(i).getAddress()))
                        {
                            flag = false;
                        }
                    }
                    if(flag == true)
                    {

detectedAdapter.add(device.getName()+"\n"+device.getAddress());
                        arrayListBluetoothDevices.add(device);
                        detectedAdapter.notifyDataSetChanged();
                    }
                }
            }
        }
    };
    private void startSearching() {
        Log.i("Log", "in the start searching method");
        IntentFilter intentFilter = new
IntentFilter(BluetoothDevice.ACTION_FOUND);
        BluetoothPair.this.registerReceiver(myReceiver, intentFilter);
        bluetoothAdapter.startDiscovery();
    }
    private void onBluetooth() {
        if(!bluetoothAdapter.isEnabled())
        {
            bluetoothAdapter.enable();
            Log.i("Log", "Bluetooth is Enabled");
        }
    }
    private void offBluetooth() {
        if(bluetoothAdapter.isEnabled())
        {
            bluetoothAdapter.disable();
        }
    }
    private void makeDiscoverable() {
        Intent discoverableIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);

discoverableIntent.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DU
RATION, 300);
        startActivity(discoverableIntent);
        Log.i("Log", "Discoverable ");
    }
    class HandleSeacrh extends Handler
    {
        @Override
        public void handleMessage(Message msg) {
            switch (msg.what) {
                case 111:


                    break;


                default:
                    break;
            }
        }
    }
}
package nsf.esarplab.scchealth;


import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.location.Address;
import android.location.Geocoder;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.wifi.WifiManager;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;


import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.LatLngBounds;


import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.json.JSONObject;


import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.List;
import java.util.Locale;


import static
nsf.esarplab.scchealth.ProfileDbHelper.CONTACTS_COLU
MN_CITY;
import static
nsf.esarplab.scchealth.ProfileDbHelper.CONTACTS_COLU
MN_EMAIL;
import static
nsf.esarplab.scchealth.ProfileDbHelper.CONTACTS_COLU
MN_ID;
import static
nsf.esarplab.scchealth.ProfileDbHelper.CONTACTS_COLU
MN_NAME;
import static
nsf.esarplab.scchealth.ProfileDbHelper.CONTACTS_COLU
MN_PHONE;
import static
nsf.esarplab.scchealth.ProfileDbHelper.CONTACTS_COLU
MN_STREET;
import static
nsf.esarplab.scchealth.ProfileDbHelper.CONTACTS_TABL
E_NAME;
```

```
public class CloudActivity extends Activity implements
View.OnClickListener {


    TextView tvIsConnected;
    EditText etName, etCountry, etdiseaseType, etTime,
etEOI, etAlg, addressET;
    Button btnPost;
    String Grid;
    Person person;
    private ProfileDbHelper mydb ;
    //private FluActivity flu;
    private String activeProfile, ptAddress, ptID;
    private String diseaseType="";
    private String eoiValue="";
    private String dateTime="";
    private String actAlg="";
    // to get long lat
    private static Context context;
    double x, y, Latitude, Longitude;
    //Button addressButton;
    TextView addressTV;
    TextView latLongTV;
    private WifiManager wifiManager;




    //String
url="http://10.100.94.221/nsf/adminlogin/insertjsondb.php";


    // String
url="https://10.100.94.221.000webhostapp.com/insertjsondb.
php";


    String url="http://sscmemphis.com/insertjsondb.php";




    public static String POST(String url, Person person)
    {
        InputStream inputStream = null;
        String result = "";
        try {


            // 1. create HttpClient
            HttpClient httpclient = new DefaultHttpClient();


            // 2. make POST request to the given URL
            HttpPost httpPost = new HttpPost(url);
            String json = "";


            // 3. build jsonObject
{"ID":"p11","GRID_CODE":"c3","DT":"BT","EOI":"5.6","T
IME":"2017-04-10"}
            // String js = etName.getText().toString();
            JSONObject jsonObject = new JSONObject();
            jsonObject.put("ID",person.getPatientID());
            jsonObject.put("GRID_CODE",
person.getGridCode());
            jsonObject.put("DT", person.getDiseaseType());
            jsonObject.put("EOI",person.getEoi());
            jsonObject.put("TIME",person.getTime());
            jsonObject.put("ALG",person.getAlgorithm());









            // 4. convert JSONObject to JSON to String
            json = jsonObject.toString();
            Log.i("Json data",json);


            // ** Alternative way to convert Person object to
JSON string usin Jackson Lib
            // ObjectMapper mapper = new ObjectMapper();
            // json = mapper.writeValueAsString(Person);


            // 5. set json to StringEntity
            StringEntity se = new StringEntity(json);


            // 6. set httpPost Entity
            httpPost.setEntity(se);


            // 7. Set some headers to inform server about the type
of the content
            httpPost.setHeader("Accept", "application/json");
            httpPost.setHeader("Content-type",
"application/json");


            // 8. Execute POST request to the given URL
            HttpResponse httpResponse =
httpclient.execute(httpPost);
```

```
            // 9. receive response as inputStream
            inputStream = httpResponse.getEntity().getContent();


            // 10. convert inputstream to string
            if (inputStream != null)
                result = convertInputStreamToString(inputStream);
            else
                result = "Did not work!";


        } catch (Exception e) {
            Log.d("InputStream", e.getLocalizedMessage());
        }


        // 11. return result
        return result;
    }


    private static String
convertInputStreamToString(InputStream inputStream)
throws IOException {
        BufferedReader bufferedReader = new
BufferedReader(new InputStreamReader(inputStream));
        String line = "";
        String result = "";
        while ((line = bufferedReader.readLine()) != null)
            result += line;


        inputStream.close();
        return result;


    }


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cloud);


        // activate wifi


        wifiManager = (WifiManager)
this.getApplicationContext().getSystemService(Context.WIFI
_SERVICE);
        if(wifiManager.isWifiEnabled()){


        }else{
            wifiManager.setWifiEnabled(true);

        }



        // get reference to the views
        tvIsConnected = (TextView)
findViewById(R.id.tvIsConnected);
        etName = (EditText) findViewById(R.id.etName);
        etCountry = (EditText) findViewById(R.id.etCountry);
        etdiseaseType = (EditText)
findViewById(R.id.etDType);
        btnPost = (Button) findViewById(R.id.btnPost);


        etTime =(EditText) findViewById(R.id.etTime);
        etEOI = (EditText) findViewById(R.id.etEOI);
        etAlg = (EditText) findViewById(R.id.etAlg);
        addressET = (EditText) findViewById(R.id.addressET);
        //flu = new FluActivity();
        //profile=flu.mNameText.getText().toString();


        // receive intent from a disease activity
        Intent diseaseIntent = getIntent();
        activeProfile=diseaseIntent.getStringExtra("profile");
        diseaseType=diseaseIntent.getStringExtra("DT");
        eoiValue=diseaseIntent.getStringExtra("EOI");
        dateTime=diseaseIntent.getStringExtra("Time");
        actAlg=diseaseIntent.getStringExtra("Algorithm");


        Log.i("profile",activeProfile);


        // fillup the fields to be shared


        // set disease type
        etdiseaseType.setText(diseaseType);


        // set patient ID
        getSubject();


        // set eoi
        etEOI.setText(eoiValue);


        // set date & time
        etTime.setText(dateTime);
        //set algorithm info
        etAlg.setText(actAlg);


        //getGridCode(address);


        // set grid code
        if(ptAddress.matches("")){
            Toast.makeText(getApplicationContext(), "No
address in database ", Toast.LENGTH_LONG).show();
        } else{
            getGridCode(ptAddress);}




        // check if you are connected or not
        if (isConnected()) {
            tvIsConnected.setBackgroundColor(0xFF00CC00);
            tvIsConnected.setText("You are connected");
        } else {
            new CountDownTimer(5000, 1000) {
                public void onFinish() {
                    // When timer is finished
                    // Execute your code here


                    if (isConnected()){

tvIsConnected.setBackgroundColor(0xFF00CC00);
                        tvIsConnected.setText("You are connected");
                    }


                    else {
                        tvIsConnected.setText("You are NOT
connected");
                    }
                }


                public void onTick(long millisUntilFinished) {
                    // millisUntilFinished    The amount of time until
finished.
                }
            }.start();




        }


        // add click listener to Button "POST"
        btnPost.setOnClickListener(this);




        // get long lat
        addressTV = (TextView)
findViewById(R.id.addressTV);
        latLongTV = (TextView)
findViewById(R.id.latLongTV);


        //addressButton = (Button)
findViewById(addressButton);


        //String address2 = addressET.getText().toString();




        /*addressButton.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View arg0) {



                // get address from database

                String address2 = addressET.getText().toString();
                getGridCode(address2);

            }
        });*/






    }


    public boolean isConnected() {
        ConnectivityManager connMgr =
(ConnectivityManager)
getSystemService(Activity.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo =
connMgr.getActiveNetworkInfo();
        if (networkInfo != null && networkInfo.isConnected())
            return true;
        else
            return false;
    }
    @Override
    public void onClick(View view) {


        switch (view.getId()) {
            case R.id.btnPost:
                if (!validate())
                    Toast.makeText(getBaseContext(), "Enter some
data!", Toast.LENGTH_LONG).show();
                // call AsynTask to perform network operation on
separate thread
                person =new Person();
                person.setPatientID(etName.getText().toString());

person.setGridCode(etCountry.getText().toString());
                person.setEoi(etEOI.getText().toString());

person.setDiseaseType(etdiseaseType.getText().toString());
                person.setTime(etTime.getText().toString());
                person.setAlgorithm(etAlg.getText().toString());
                new HttpAsyncTask().execute(url);
                finish();
                break;
        }


    }


    private boolean validate() {
        if (etName.getText().toString().trim().equals(""))
            return false;
        else if (etCountry.getText().toString().trim().equals(""))
            return false;
        else if
(etdiseaseType.getText().toString().trim().equals(""))
            return false;
        else
            return true;
    }


    private class HttpAsyncTask extends AsyncTask<String,
Void, String> {
        @Override
        protected String doInBackground(String... urls) {


           /* Person = new Person();
            Person.setName(etName.getText().toString());
            Person.setCountry(etCountry.getText().toString());
            Person.setTwitter(etTwitter.getText().toString());*/


            return POST(urls[0], person);
        }


        // onPostExecute displays the results of the AsyncTask.
        @Override
        protected void onPostExecute(String result) {
            Toast.makeText(getBaseContext(), "Data Sent!",
Toast.LENGTH_LONG).show();
        }
    }


    // long lat


    public void getGridCode(String addressinput){


        String locationName=addressinput;


        Geocoder geoCoder = new Geocoder(this,
Locale.ENGLISH);
        try {
            List<Address> address =
geoCoder.getFromLocationName(locationName, 1);
            Latitude = address.get(0).getLatitude();
            Longitude = address.get(0).getLongitude();
            Log.i("Lat", "" + Latitude);
            Log.i("Lng", "" + Longitude);
        } catch (IOException e) {
            e.printStackTrace();
            Log.i("INFO", "exception");
        }
        x = Longitude;// -89.952302;
        y = Latitude;//35.1149703;


        LatLngBounds AZ19 = new LatLngBounds(
                new LatLng(-90.085643, 34.922824), new
LatLng(-90.085642, 35.0317400825));
        LatLngBounds ZG86 = new LatLngBounds(
                new LatLng(-90.085642, 34.922824), new
LatLng(-89.861741, 35.0317400825));
        LatLngBounds XP52 = new LatLngBounds(
                new LatLng(-89.8617410, 34.922824), new
LatLng(-89.6378400, 35.0317400825));
        LatLngBounds DW46 = new LatLngBounds(
                new LatLng(-89.637840, 34.922824), new
LatLng(-89.413939, 35.0317400825));


        LatLngBounds FD32 = new LatLngBounds(
                new LatLng(-90.085643, 35.0317400825), new
LatLng(-90.085642, 35.140656165));
        //central area
        LatLngBounds YU76B = new LatLngBounds(
                new LatLng(-90.085642, 35.08619812), new
LatLng(-89.9736915, 35.140656165));
        LatLngBounds YU76L = new LatLngBounds(
                new LatLng(-90.085642, 35.0317400825), new
LatLng(-89.9736915, 35.08619812));
        LatLngBounds YU76K = new LatLngBounds(
                new LatLng(-89.9736915, 35.08619812), new
LatLng(-89.861741, 35.140656165));
        LatLngBounds YU76Z = new LatLngBounds(
                new LatLng(-89.9736915, 35.0317400825), new
LatLng(-89.861741, 35.08619812));
        //central end
        LatLngBounds HD93 = new LatLngBounds(
                new LatLng(-89.861741, 35.0317400825), new
LatLng(-89.637840, 35.140656165));
        LatLngBounds WG49 = new LatLngBounds(
                new LatLng(-89.637840,35.0317400825), new
LatLng(-89.413939, 35.140656165));


        LatLngBounds SP71 = new LatLngBounds(
                new LatLng(-90.309543, 35.140656165), new
LatLng(-90.085642, 35.2495722475));
        LatLngBounds KT43 = new LatLngBounds(
                new LatLng(-90.085642, 35.140656165), new
LatLng(-89.861741, 35.2495722475));
        LatLngBounds BY28 = new LatLngBounds(
                new LatLng(-89.861741, 35.140656165), new
LatLng(-89.637840, 35.2495722475));
        LatLngBounds LC95 = new LatLngBounds(
                new LatLng(-89.637840, 35.140656165), new
LatLng(-89.413939, 35.2495722475));


        LatLngBounds CR63 = new LatLngBounds(
                new LatLng(-90.309543, 35.2495722475), new
LatLng(-90.085642, 35.35848833));
        LatLngBounds CX38 = new LatLngBounds(
                new LatLng(-90.085642, 35.2495722475), new
LatLng(-89.861741, 35.35848833));
        LatLngBounds VJ14 = new LatLngBounds(
                new LatLng(-89.861741, 35.2495722475), new
LatLng(-89.637840, 35.35848833));
        LatLngBounds DR27 = new LatLngBounds(
                new LatLng(-89.637840, 35.2495722475), new
LatLng(-89.413939, 35.35848833));




        if (AZ19.contains(new LatLng(x, y))) {
            Log.i("Grid Code", "AZ19");
            Grid="AZ19";
        } else if (ZG86.contains(new LatLng(x, y))) {
            Log.i("Grid Code", "ZG86");
```

```
    Grid="ZG86";
  } else if (BY28.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "BY28");
    Grid="BY28";
  } else if (XP52.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "XP52");
    Grid="XP52";
  } else if (DW46.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "DW46");
    Grid="DW46";
  } else if (FD32.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "FD32");
    Grid="FD32";
  } else if (BY28.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "BY28");
    Grid="BY28";
  } else if (YU76B.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "YU76B");
    Grid="YU76B";
  } else if (YU76L.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "YU76L");
    Grid="YU76L";
  } else if (YU76K.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "YU76K");
    Grid="YU76K";
  }else if (YU76Z .contains(new LatLng(x, y))) {
    Log.i("Grid Code", "YU76Z");
    Grid="YU76Z";
  } else if (HD93.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "HD93");
    Grid="HD93";
  } else if (WG49.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "WG49");
    Grid="WG49";
  }  else if (SP71.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "SP71");
    Grid="SP71";
  }else if (KT43.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "KT43");
    Grid="KT43";
  }else if (BY28.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "BY28");
    Grid="BY28";
  } else if (LC95.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "LC95");
    Grid="LC95";
  } else if (CR63.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "CR63");
    Grid="CR63";
  } else if (CX38.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "CX38");
    Grid="CX38";
  } else if (VJ14.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "VJ14");
    Grid="VJ14";
  } else if (DR27.contains(new LatLng(x, y))) {
```

```
            Log.i("Grid Code", "DR27");
            Grid="DR27";
        } else {
            Log.i("Lat2", "" + x);
            Log.i("Lng2", "" + y);
            Log.i("Grid Code", "Outside Boundary");
            Grid="Outside Boundary";
        }
        etCountry.setText(Grid);
    }



    public void getSubject() throws SQLException{


        try {
            mydb = new ProfileDbHelper(this);
            SQLiteDatabase ourDatabase =
mydb.getReadableDatabase();
            String[] columns = new
String[]{CONTACTS_COLUMN_ID,
CONTACTS_COLUMN_NAME,
CONTACTS_COLUMN_EMAIL,
CONTACTS_COLUMN_STREET,
CONTACTS_COLUMN_CITY,
CONTACTS_COLUMN_PHONE};
            Cursor c =
ourDatabase.query(CONTACTS_TABLE_NAME,
columns,null , null, null, null, null);
            ptAddress = "";
            ptID="";


            // patient id
            int iRow=
c.getColumnIndex(CONTACTS_COLUMN_PHONE);
            //int iName=
c.getColumnIndex(CONTACTS_COLUMN_NAME);
            int iAddress=
c.getColumnIndex(CONTACTS_COLUMN_STREET);
        /*for (c.moveToFirst();
!c.isAfterLast();c.moveToNext()){
            result = result + c.getString(iAddress);
            ptID = ptID + c.getString(iName);
        }*/
       /* if(c!=null)
        {c.moveToFirst();
            ptID=c.getString(1);}
         etName.setText(ptID);*/
            c.moveToFirst();
            do {
                if ((c.getString(1)).equals(activeProfile)) {
                    ptID = ptID+c.getString(iRow);
                    ptAddress = ptAddress + c.getString(iAddress);
                    }


                } while (c.moveToNext());
                etName.setText(ptID);
                //etCountry.setText(result);
                addressET.setText(ptAddress);
                //return result;
            }


            catch(Exception e){
                etName.setText("");
                //etCountry.setText(result);
                addressET.setText("");
            }
        }
    }
package nsf.esarplab.scchealth;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;


public class ConnectionSettingsActivity extends Activity {


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_connection_settings);




        // Find the View that shows the profile category
        TextView bluetooth = (TextView)
findViewById(R.id.bluetooth);


        // Set a click listener on that View
        bluetooth.setOnClickListener(new View.OnClickListener()
{
            // The code in this method will be executed when the
profile category is clicked on.
            @Override
            public void onClick(View view) {
                // Create a new intent to open the {@link
ProfileActivity}
                Intent bluetoothIntent = new
Intent(ConnectionSettingsActivity.this, BluetoothActivity.class);


                // Start the new activity
                startActivity(bluetoothIntent);


            }
        });


    /*   // Find the View that shows the profile category
        TextView wifi = (TextView) findViewById(R.id.wifi);
        // Set a click listener on that View
        wifi.setOnClickListener(new View.OnClickListener() {
            // The code in this method will be executed when the
profile category is clicked on.
            @Override
            public void onClick(View view) {
                // Create a new intent to open the {@link
ProfileActivity}
                Intent wifiIntent = new
Intent(ConnectionSettingsActivity.this, WifiActivity.class);

                // Start the new activity
                startActivity(wifiIntent);

            }
        });*/
    }
}
package nsf.esarplab.scchealth;


import android.content.Context;


import java.io.File;
import java.util.concurrent.ExecutionException;


import dalvik.system.DexClassLoader;


/**
 * Created by mrahman8 on 6/8/2017.
 */


public class CustomizedDexClassLoader {


    private static Context context;


    private static DexClassLoader loader;


    public static void setContext(Context context) {
        CustomizedDexClassLoader.context = context;
    }


    public static DexClassLoader load(final String dexFileName) throws
RuntimeException {
        if(null == context) {
            throw new RuntimeException("No context provided");
        }
        if(null == loader) {
            final File dexInternalStoragePath = new File(context.getDir("dex",
Context.MODE_PRIVATE), dexFileName);
            if (!dexInternalStoragePath.exists()) {
                try {
                    (new DexPreparationTask(context,
dexFileName)).execute(dexInternalStoragePath).get();
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                } catch (ExecutionException e) {
                    throw new RuntimeException(e);
                }
            }
            final File optimizedDexOutputPath = context.getDir("outdex",
Context.MODE_PRIVATE);


            loader = new
DexClassLoader(dexInternalStoragePath.getAbsolutePath(),
optimizedDexOutputPath.getAbsolutePath(), null,
context.getClassLoader().getParent());
        }
        return loader;
    }


}
package nsf.esarplab.scchealth;


            import android.content.Intent;
            import android.graphics.Color;
            import android.os.Bundle;
            import android.support.v7.app.ActionBar;
            import android.support.v7.app.AppCompatActivity;
            import android.view.View;
            import android.widget.Button;
            import android.widget.EditText;
            import android.widget.TextView;
            import android.widget.Toast;


            public class DB_login extends AppCompatActivity {
                Button b1,b2;
                EditText ed1,ed2;


                TextView tx1, tx2;
```

```
  int counter = 3;


  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_db_login);
    // show action bar
    ActionBar myActionBar = getSupportActionBar();
    myActionBar.show();


    b1 = (Button)findViewById(R.id.button);
    ed1 = (EditText)findViewById(R.id.editText);
    ed2 = (EditText)findViewById(R.id.editText2);


    b2 = (Button)findViewById(R.id.button2);
    tx1 = (TextView)findViewById(R.id.textView3);
    tx2 = (TextView)findViewById(R.id.textView2);
    tx1.setVisibility(View.GONE);
    tx2.setVisibility(View.GONE);


    b1.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        if(ed1.getText().toString().equals("admin") &&
            ed2.getText().toString().equals("hce")) {


          Intent NewActivityIntent = new Intent(DB_login.this,
GeneralSettingsActivity.class);


          // Start the new activity
          startActivity(NewActivityIntent);


            /* Toast.makeText(getApplicationContext(),

"Redirecting...",Toast.LENGTH_SHORT).show();*/
        }
        else{
          Toast.makeText(getApplicationContext(), "Wrong
Credentials",Toast.LENGTH_SHORT).show();


          tx1.setVisibility(View.VISIBLE);
          tx2.setVisibility(View.VISIBLE);
          tx1.setBackgroundColor(Color.RED);
          counter--;
          tx1.setText(Integer.toString(counter));


          if (counter == 0) {
```

```
                    b1.setEnabled(false);
                }
            }
        }
    });


    b2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            finish();
        }
    });
  }
}
package nsf.esarplab.scchealth;



import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;


/**
 * Created by mrahman8 on 7/24/2017.
 */


public class DataBaseHelper extends SQLiteOpenHelper {


   public DataBaseHelper(Context context, String name,
SQLiteDatabase.CursorFactory factory, int version)
   {
      super(context, name, factory, version);
   }
   // Called when no database exists in disk and the helper class needs
   // to create a new one.
   @Override
   public void onCreate(SQLiteDatabase _db)
   {
      _db.execSQL(LoginDataBaseAdapter.DATABASE_CREATE);


   }
   // Called when there is a database version mismatch meaning that the
version
   // of the database on disk needs to be upgraded to the current version.
   @Override
   public void onUpgrade(SQLiteDatabase _db, int _oldVersion, int
_newVersion)
   {
      // Log the version upgrade.
```

```
                                        Log.w("TaskDBAdapter", "Upgrading from version " +_oldVersion
                                    + " to " +_newVersion + ", which will destroy all old data");




                                        // Upgrade the existing database to conform to the new version.
                                    Multiple
                                        // previous versions can be handled by comparing _oldVersion and
                                    _newVersion
                                        // values.
                                        // The simplest case is to drop the old table and create a new one.
                                        _db.execSQL("DROP TABLE IF EXISTS " + "TEMPLATE");
                                        // Create a new one.
                                        onCreate(_db);
                                      }
                                    }
package nsf.esarplab.scchealth;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.EditText;


public class GraphActivity extends Activity {

  EditText num1, num2, num3, num4, num5;
  Button btnShow;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_graph);
    getWindow().setSoftInputMode(
          WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWAYS_HIDDEN
    );

   /* num1 = (EditText)findViewById(R.id.num1);
    num2 = (EditText)findViewById(R.id.num2);
    num3 = (EditText)findViewById(R.id.num3);
    num4 = (EditText)findViewById(R.id.num4);
    num5 = (EditText)findViewById(R.id.num5);*/
    btnShow = (Button)findViewById(R.id.show);

    btnShow.setOnClickListener(btnShowOnClickListener);
  }

  OnClickListener btnShowOnClickListener =
      new OnClickListener(){

        @Override
```

```
            public void onClick(View v) {
              Intent intent = new Intent(
                  GraphActivity.this,
                  ShowWebChart.class);

             /* intent.putExtra("NUM1", getNum(num1));
              intent.putExtra("NUM2", getNum(num2));
              intent.putExtra("NUM3", getNum(num3));
              intent.putExtra("NUM4", getNum(num4));
              intent.putExtra("NUM5", getNum(num5));
*/

              intent.putExtra("NUM1", 40);
              intent.putExtra("NUM2", 50);
              intent.putExtra("NUM3", 70);
              intent.putExtra("NUM4", 35);
              intent.putExtra("NUM5", 50);

              startActivity(intent);
            }

          };

  private int getNum(EditText editText){

    int num = 0;

    String stringNum = editText.getText().toString();
    if(!stringNum.equals("")){
      num = Integer.valueOf(stringNum);
    }

    return (num);
  }

}
package nsf.esarplab.scchealth;

import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.support.v7.app.NotificationCompat;

/**
 * Created by mrahman8 on 7/25/2017.
 */

public class Notification_receiver extends BroadcastReceiver {

  @Override
  public void onReceive(Context context, Intent intent){

    NotificationManager notificationManager=(NotificationManager)
context.getSystemService(Context.NOTIFICATION_SERVICE);
```

```
        Intent repeating_intent=new Intent(context,LoginActivity.class);

        PendingIntent
pendingIntent=PendingIntent.getActivity(context,100,repeating_intent,PendingIntent.FLAG_UPDATE_CURRENT
) ;

        NotificationCompat.Builder builder=(android.support.v7.app.NotificationCompat.Builder) new
NotificationCompat.Builder(context)
                .setContentIntent(pendingIntent)
                .setSmallIcon(R.drawable.flogo)
                .setContentTitle("Health Checkup Reminder")
                .setContentText("Hey! It's time to do the test")
                .setAutoCancel(true);


        notificationManager.notify(100, builder.build());
    }


}
package nsf.esarplab.scchealth;

import android.app.Activity;
import android.content.Context;
import android.location.Address;
import android.location.Geocoder;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.LatLngBounds;

import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.text.DateFormat;
import java.util.Date;
import java.util.List;
import java.util.Locale;

public class WifiActivity extends AppCompatActivity implements View.OnClickListener {

    TextView tvIsConnected;
    EditText etName, etCountry, etdiseaseType, etTime, etEOI, etAlg;
    Button btnPost;
    String Grid;
    Person person;

    // to get long lat
    private static Context context;
    double x, y, Latitude,Longitude;
    Button addressButton;
    TextView addressTV;
    TextView latLongTV;

    //String url="http://10.100.94.221/nsf/adminlogin/insertjsondb.php";

    // String url="https://10.100.94.221.000webhostapp.com/insertjsondb.php";

    String url="http://sscmemphis.com/insertjsondb.php";

    public static String POST(String url, Person person)
    {
        InputStream inputStream = null;
        String result = "";
        try {

            // 1. create HttpClient
            HttpClient httpclient = new DefaultHttpClient();

            // 2. make POST request to the given URL
            HttpPost httpPost = new HttpPost(url);

            String json = "";

            // 3. build jsonObject {"ID":"p11","GRID_CODE":"c3","DT":"BT","EOI":"5.6","TIME":"2017-04-10"}
            // String js = etName.getText().toString();
            JSONObject jsonObject = new JSONObject();
            jsonObject.put("ID",person.getPatientID());
            jsonObject.put("GRID_CODE", person.getGridCode());
            jsonObject.put("DT", person.getDiseaseType());
            jsonObject.put("EOI",person.getEoi());
            jsonObject.put("TIME",person.getTime());
            jsonObject.put("ALG","BT1");
            //jsonObject.put("ALG",person.getAlgorithm());




            // 4. convert JSONObject to JSON to String
            json = jsonObject.toString();
            Log.i("Json data",json);
            // ** Alternative way to convert Person object to JSON string usin Jackson Lib
            // ObjectMapper mapper = new ObjectMapper();
            // json = mapper.writeValueAsString(Person);

            // 5. set json to StringEntity
            StringEntity se = new StringEntity(json);

            // 6. set httpPost Entity
            httpPost.setEntity(se);

            // 7. Set some headers to inform server about the type of the content
            httpPost.setHeader("Accept", "application/json");
            httpPost.setHeader("Content-type", "application/json");

            // 8. Execute POST request to the given URL
            HttpResponse httpResponse = httpclient.execute(httpPost);

            // 9. receive response as inputStream
            inputStream = httpResponse.getEntity().getContent();

            // 10. convert inputstream to string
            if (inputStream != null)
                result = convertInputStreamToString(inputStream);
            else
                result = "Did not work!";

        } catch (Exception e) {
            Log.d("InputStream", e.getLocalizedMessage());
        }

        // 11. return result
        return result;
    }

    private static String convertInputStreamToString(InputStream inputStream) throws IOException {
        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream));
        String line = "";
        String result = "";
        while ((line = bufferedReader.readLine()) != null)
            result += line;

        inputStream.close();
        return result;

    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_wifi);

        // show action bar
        ActionBar myActionBar = getSupportActionBar();
        myActionBar.show();

        // get reference to the views
        tvIsConnected = (TextView) findViewById(R.id.tvIsConnected);
        etName = (EditText) findViewById(R.id.etName);
        etCountry = (EditText) findViewById(R.id.etCountry);
        etdiseaseType = (EditText) findViewById(R.id.etDType);
        btnPost = (Button) findViewById(R.id.btnPost);

        etTime =(EditText) findViewById(R.id.etTime);
        etEOI = (EditText) findViewById(R.id.etEOI);
        etAlg = (EditText) findViewById(R.id.etAlg);
        // set date time
        String currentDateTime = DateFormat.getDateTimeInstance().format(new Date());
        etTime.setText(currentDateTime);

        // check if you are connected or not
        if (isConnected()) {
            tvIsConnected.setBackgroundColor(0xFF00CC00);
            tvIsConnected.setText("You are conncted");
        } else {
            tvIsConnected.setText("You are NOT conncted");
        }

        // add click listener to Button "POST"
        btnPost.setOnClickListener(this);


        // get long lat
        addressTV = (TextView) findViewById(R.id.addressTV);
        latLongTV = (TextView) findViewById(R.id.latLongTV);

        addressButton = (Button) findViewById(R.id.addressButton);
        addressButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View arg0) {

                EditText editText = (EditText) findViewById(R.id.addressET);
                String address = editText.getText().toString();
                getGridCode(address);
            }
        });



    }

    public boolean isConnected() {
        ConnectivityManager connMgr = (ConnectivityManager)
getSystemService(Activity.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
        if (networkInfo != null && networkInfo.isConnected())
            return true;
        else
            return false;
    }

    @Override
    public void onClick(View view) {
```

```
    switch (view.getId()) {
      case R.id.btnPost:
        if (!validate())
          Toast.makeText(getBaseContext(), "Enter some data!", Toast.LENGTH_LONG).show();
        // call AsynTask to perform network operation on separate thread


        person =new Person();
        person.setPatientID(etName.getText().toString());
        person.setGridCode(etCountry.getText().toString());
        person.setEoi(etEOI.getText().toString());
        person.setDiseaseType(etdiseaseType.getText().toString());
        person.setTime(etTime.getText().toString());
        person.setAlgorithm(etAlg.getText().toString());
        new HttpAsyncTask().execute(url);


        break;
    }

  }

  private boolean validate() {
    if (etName.getText().toString().trim().equals(""))
      return false;
    else if (etCountry.getText().toString().trim().equals(""))
      return false;
    else if (etdiseaseType.getText().toString().trim().equals(""))
      return false;
    else
      return true;
  }

  private class HttpAsyncTask extends AsyncTask<String, Void, String> {
    @Override
    protected String doInBackground(String... urls) {

      /* Person = new Person();
       Person.setName(etName.getText().toString());
       Person.setCountry(etCountry.getText().toString());
       Person.setTwitter(etTwitter.getText().toString());*/

      return POST(urls[0], person);
    }

    // onPostExecute displays the results of the AsyncTask.
    @Override
    protected void onPostExecute(String result) {
      Toast.makeText(getBaseContext(), "Data Sent!", Toast.LENGTH_LONG).show();
    }
  }

  // long lat

  public void getGridCode(String addressinput){
```

```
String locationName=addressinput;

Geocoder geoCoder = new Geocoder(this, Locale.ENGLISH);
try {
   List<Address> address = geoCoder.getFromLocationName(locationName, 1);
   Latitude = address.get(0).getLatitude();
   Longitude = address.get(0).getLongitude();
   Log.i("Lat", "" + Latitude);
   Log.i("Lng", "" + Longitude);
} catch (IOException e) {
   e.printStackTrace();
   Log.i("INFO", "exception");
}
x = Longitude;// -89.952302;
y = Latitude;//35.1149703;

LatLngBounds AZ19 = new LatLngBounds(
       new LatLng(-90.085643, 34.922824), new LatLng(-90.085642, 35.0317400825));
LatLngBounds ZG86 = new LatLngBounds(
       new LatLng(-90.085642, 34.922824), new LatLng(-89.861741, 35.0317400825));
LatLngBounds XP52 = new LatLngBounds(
       new LatLng(-89.8617410, 34.922824), new LatLng(-89.6378400, 35.0317400825));
LatLngBounds DW46 = new LatLngBounds(
       new LatLng(-89.637840, 34.922824), new LatLng(-89.413939, 35.0317400825));

LatLngBounds FD32 = new LatLngBounds(
       new LatLng(-90.085643, 35.0317400825), new LatLng(-90.085642, 35.140656165));
//central area
LatLngBounds YU76B = new LatLngBounds(
       new LatLng(-90.085642, 35.08619812), new LatLng(-89.9736915, 35.140656165));
LatLngBounds YU76L = new LatLngBounds(
       new LatLng(-90.085642, 35.0317400825), new LatLng(-89.9736915, 35.08619812));
LatLngBounds YU76K = new LatLngBounds(
       new LatLng(-89.9736915, 35.08619812), new LatLng(-89.861741, 35.140656165));
LatLngBounds YU76Z = new LatLngBounds(
       new LatLng(-89.9736915, 35.0317400825), new LatLng(-89.861741, 35.08619812));
//central end
LatLngBounds HD93 = new LatLngBounds(
       new LatLng(-89.861741, 35.0317400825), new LatLng(-89.637840, 35.140656165));
LatLngBounds WG49 = new LatLngBounds(
       new LatLng(-89.637840,35.0317400825), new LatLng(-89.413939, 35.140656165));

LatLngBounds SP71 = new LatLngBounds(
       new LatLng(-90.309543, 35.140656165), new LatLng(-90.085642, 35.2495722475));
LatLngBounds KT43 = new LatLngBounds(
       new LatLng(-90.085642, 35.140656165), new LatLng(-89.861741, 35.2495722475));
LatLngBounds BY28 = new LatLngBounds(
       new LatLng(-89.861741, 35.140656165), new LatLng(-89.637840, 35.2495722475));
LatLngBounds LC95 = new LatLngBounds(
       new LatLng(-89.637840, 35.140656165), new LatLng(-89.413939, 35.2495722475));

LatLngBounds CR63 = new LatLngBounds(
       new LatLng(-90.309543, 35.2495722475), new LatLng(-90.085642, 35.35848833));
LatLngBounds CX38 = new LatLngBounds(
       new LatLng(-90.085642, 35.2495722475), new LatLng(-89.861741, 35.35848833));
```

```
LatLngBounds VJ14 = new LatLngBounds(
        new LatLng(-89.861741, 35.2495722475), new LatLng(-89.637840, 35.35848833));
LatLngBounds DR27 = new LatLngBounds(
        new LatLng(-89.637840, 35.2495722475), new LatLng(-89.413939, 35.35848833));


if (AZ19.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "AZ19");
    Grid="AZ19";
} else if (ZG86.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "ZG86");
    Grid="ZG86";
} else if (BY28.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "BY28");
    Grid="BY28";
} else if (XP52.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "XP52");
    Grid="XP52";
} else if (DW46.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "DW46");
    Grid="DW46";
} else if (FD32.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "FD32");
    Grid="FD32";
} else if (BY28.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "BY28");
    Grid="BY28";
} else if (YU76B.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "YU76B");
    Grid="YU76B";
} else if (YU76L.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "YU76L");
    Grid="YU76L";
} else if (YU76K.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "YU76K");
    Grid="YU76K";
}else if (YU76Z .contains(new LatLng(x, y))) {
    Log.i("Grid Code", "YU76Z");
    Grid="YU76Z";
} else if (HD93.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "HD93");
    Grid="HD93";
} else if (WG49.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "WG49");
    Grid="WG49";
}  else if (SP71.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "SP71");
    Grid="SP71";
}else if (KT43.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "KT43");
    Grid="KT43";
}else if (BY28.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "BY28");
    Grid="BY28";
} else if (LC95.contains(new LatLng(x, y))) {
    Log.i("Grid Code", "LC95");
```

```
            Grid="LC95";
        } else if (CR63.contains(new LatLng(x, y))) {
            Log.i("Grid Code", "CR63");
            Grid="CR63";
        } else if (CX38.contains(new LatLng(x, y))) {
            Log.i("Grid Code", "CX38");
            Grid="CX38";
        } else if (VJ14.contains(new LatLng(x, y))) {
            Log.i("Grid Code", "VJ14");
            Grid="VJ14";
        } else if (DR27.contains(new LatLng(x, y))) {
            Log.i("Grid Code", "DR27");
            Grid="DR27";
        } else {
            Log.i("Lat2", "" + x);
            Log.i("Lng2", "" + y);
            Log.i("Grid Code", "Outside Boundary");
            Grid="Outside Boundary";
        }
        latLongTV.setText("Grid Code:"+Grid);
    }
}
package nsf.esarplab.scchealth;

import android.app.Activity;
import android.app.ProgressDialog;
import android.bluetooth.BluetoothAdapter;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.graphics.drawable.GradientDrawable;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.os.Environment;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.telephony.SmsManager;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.PopupWindow;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.ScrollView;
import android.widget.TextView;
import android.widget.Toast;
import com.google.android.gms.common.api.GoogleApiClient;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.text.DateFormat;
import java.text.DecimalFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.Locale;

import nsf.esarplab.bluetoothlibrary.BluetoothSPP;
import nsf.esarplab.bluetoothlibrary.BluetoothState;
import nsf.esarplab.bluetoothlibrary.DeviceList;

public class oneStopService extends AppCompatActivity {

    BluetoothSPP bt;
    private TextView intro, btData, brData,hrData, ecData, oximetryData, connectionRead;
    private TextView statusTemp, statusHR, statusBR, statusO2, statusECG,
gradientFlu,gradientSA,gradientAR,gradientPD, fluScreening,postStatus;
    private LinearLayout layout1,layoutIntro,layoutProfile,sensorDisplay,sensorStatus;
    private ScrollView  mDisplay;
    EditText etMessage;
    private GoogleApiClient client;
    private Button connectScanner, collectData, dispResult, shareResult;
    private boolean diseaseKey = false;
    private boolean sensorKeyBT = false;
    private boolean sensorKeyPO = false;
    private boolean sensorKeyHR = false;
    private boolean sensorKeyBR = false;
    private boolean sensorKeyEC = false;
    private boolean timeKeyBT = false;
    private boolean timeKeyPO = false;
    private boolean timeKeyHR = false;
    private boolean timeKeyBR = false;
    private boolean timeKeyEC=false;
    private boolean handShake=false;
    private int sensor = 1;
    private int fileSeq=1;
    private int sensorNo;
    private String eoiValue,sSeverity;
    double ratingOfEOI = 0.0;
    private  RadioGroup radioGroup;
    private String sEOI="";
    private String sTemperature="";
    private RadioButton rb1, rb2, rb3, rb4, rb5;
    Menu menu;
    private ArrayList<String> arr_hex = new ArrayList<String>();
    private ArrayList<Short> arr_received = new ArrayList<Short>();
    private ArrayList<Short> arr_respiration = new ArrayList<Short>();
    private String currentDateTime = "";
    String s = "";
```

```
private PopupWindow mPopupWindow;
private CheckBox ch1,ch2;
private double temperature=0.0f;
private ProgressDialog progressDialog;
private CountDownTimer Count;
private boolean failedHandshake;


@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_one_stop_service);

    // show action bar
    ActionBar myActionBar = getSupportActionBar();
    myActionBar.show();
    TextView mNameText = (TextView) findViewById(R.id.display_name);
    bt = new BluetoothSPP(this);
    // initialize layouts
    layout1=(LinearLayout)findViewById(R.id.layout1);
    layoutIntro=(LinearLayout)findViewById(R.id.layout_intro);
    layoutProfile=(LinearLayout)findViewById(R.id.layout2);
    sensorStatus=(LinearLayout) findViewById(R.id.sensorStatus);

    //intro=(TextView) findViewById(R.id.display_intro);
    sensorDisplay=(LinearLayout) findViewById(R.id.sensorDisplay);
    mDisplay=(ScrollView) findViewById(R.id.maindisp);
    collectData=(Button) findViewById(R.id.test);
    btData=(TextView) findViewById(R.id.value_flu);

    // sensor status
    statusTemp=(TextView) findViewById(R.id.statusTemp);
    statusHR=(TextView) findViewById(R.id.statusHR);
    statusBR=(TextView) findViewById(R.id.statusBR);
    //statusO2=(TextView) findViewById(R.id.statusO2);
    statusECG=(TextView) findViewById(R.id.statusECG);
    fluScreening=(TextView) findViewById(R.id.screening_flu);
    postStatus=(TextView) findViewById(R.id.postStatus);

    oximetryData=(TextView) findViewById(R.id.display_spo2);
    connectionRead = (TextView) findViewById(R.id.textStatus);
    dispResult = (Button) findViewById(R.id.displayResult);
    connectScanner = (Button) findViewById(R.id.cScanner);
    shareResult=(Button) findViewById(R.id.shareResult);
    sensorNo=0;

    // hide sensor list & collect data button
    sensorDisplay.setVisibility(View.GONE);
    collectData.setVisibility(View.GONE);
    // hide compute & share button
    dispResult.setVisibility(View.GONE);
    shareResult.setVisibility(View.GONE);

    //Show active profile
    SharedPreferences prefs = getSharedPreferences("logindetails",MODE_PRIVATE);
    String Uname =  prefs.getString("loginname","Default");
```

```
        mNameText.setText("\t\t"+Uname);

        s = mNameText.getText().toString().trim();

        // color line gradient


        gradientFlu = (TextView) findViewById(R.id.active_gradient_flu);
        gradientSA = (TextView) findViewById(R.id.active_gradient_SA);
        gradientAR = (TextView) findViewById(R.id.active_gradient_AR);
        gradientPD = (TextView) findViewById(R.id.active_gradient_PD);

        int[] colors = {Color.parseColor("#008000"), Color.parseColor("#FFFF00"), Color.parseColor("#FFA500"),
Color.parseColor("#ff0000"), Color.parseColor("#800000")};
        GradientDrawable gd = new GradientDrawable(
                GradientDrawable.Orientation.LEFT_RIGHT, colors);
        gradientFlu.setBackground(gd);
        gradientSA.setBackground(gd);
        gradientAR.setBackground(gd);
        gradientPD.setBackground(gd);

        // radio group for sensors

        radioGroup = (RadioGroup) findViewById(R.id.rg1);
        rb1 = (RadioButton) findViewById(R.id.rb_bt);
        //rb2 = (RadioButton) findViewById(R.id.rb_o2);
        //rb3 = (RadioButton) findViewById(R.id.rb_hr);
        //rb4 = (RadioButton) findViewById(R.id.rb_resp);
        //rb5 = (RadioButton) findViewById(R.id.rb_ecg);
        rb1.setTextColor(Color.BLUE);



        radioGroup.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener()
        {
            @Override
            public void onCheckedChanged(RadioGroup group, int checkedId) {

                // set text color
                int selectedId = radioGroup.getCheckedRadioButtonId();
                RadioButton selected = (RadioButton) findViewById(selectedId);
                rb1.setTextColor(Color.BLACK);
                //rb2.setTextColor(Color.BLACK);
                //rb3.setTextColor(Color.BLACK);
                //rb4.setTextColor(Color.BLACK);
                //rb5.setTextColor(Color.BLACK);

                selected.setTextColor(Color.BLUE);
                // checkedId is the RadioButton selected
                if(checkedId==R.id.rb_bt){

                    sensor=1;
                }else if(checkedId==R.id.rb_resp){

                    sensor=2;
                }
        else if(checkedId==R.id.rb_o2){

            sensor=3;
        }else if(checkedId==R.id.rb_hr){

            sensor=4;
        }else if(checkedId==R.id.rb_ecg){

            sensor=5;
        }

    }
});


// set the bluetooth connection

bt.setOnDataReceivedListener(new BluetoothSPP.OnDataReceivedListener() {
    public void onDataReceived(byte[] data, String message) {
        short val = 0;
        String readAscii = new String(data);
        //textReceived.append(message + "\n");
        if (handShake) {

            arr_hex.add(message);
            Log.i("Str2@activity",message);
            Log.i("size_arr_hex",""+arr_hex.size());


            if (arr_hex.size() == 2) {
                String catHex = arr_hex.get(0) + arr_hex.get(1);
                /*int b0 = (arr_hex.get(0) & 255); // converts to unsigned
                int b1 = (arr_hex.get(1) & 255); // converts to unsigned
                int val = b0 << 8 | b1;*/
                Log.i("val1@final", catHex);
                //val = (short) (Integer.parseInt(catHex, 16));
                //Log.i("val2@final", String.valueOf(val));

                //val= (short) Long.parseLong(catHex, 16);
                val = (short) (Integer.parseInt(catHex, 16));

                //val= (int) Long.parseLong(catHex, 16);
                Log.i("val3@final", String.valueOf(val));



                arr_received.add(val);
                //Textv.append(Integer.toString(val) + "\n");
                try {
                    writeToCsv(Integer.toString(val));
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                arr_hex.clear();
        }
        //receive data



        /*switch (sensor) {
            case 1: {
                arr_received.add(val);
                Log.i("sizearr_received", "" + arr_received.size());
                break;
            }
            case 2: {
                arr_respiration.add(val);
                //Log.i("sizearr_received", "" + arr_received.size());
                break;
            }
            case 3: {
                //hrData.append(message + "\n");
                break;
            }
            case 4: {
                //brData.append(message + "\n");
                break;
            }
            case 5: {
                //ecData.append(message + "\n");
                break;
            }
        }*/

    } else {
        if (readAscii.equals("OS")) {

            switch (sensor) {
                case 1: {
                    bt.send("TP", true);
                    diseaseKey = true;
                    break;
                }
                case 2: {
                    bt.send("PO", true);
                    diseaseKey = true;
                    break;
                }
                case 3: {
                    bt.send("HR", true);
                    diseaseKey = true;
                    break;
                }
                case 4: {
                    bt.send("BR", true);
                    diseaseKey = true;
                    break;
                }
                case 5: {
                    bt.send("EC", true);
                    diseaseKey = true;
                    break;
                }
            }

        } else {
            if (readAscii.equals("TP") && diseaseKey) {
                bt.send("00020", true);
                sensorKeyBT = true;
            } else if (readAscii.equals("PO") && diseaseKey) {
                bt.send("5", true);
                sensorKeyPO= true;
            } else if (readAscii.equals("HR") && diseaseKey) {
                bt.send("6", true);
                sensorKeyHR = true;
            }else if (readAscii.equals("BR") && diseaseKey) {
                bt.send("7", true);
                sensorKeyBR = true;
            }else if (readAscii.equals("EC") && diseaseKey) {
                bt.send("8", true);
                sensorKeyEC = true;
            }else {
                if (readAscii.equals("00020") && sensorKeyBT && diseaseKey) {
                    bt.send("OK", true);
                    handShake = true;
                } else if (readAscii.equals("5") && sensorKeyPO && diseaseKey) {
                    bt.send("OK", true);
                    handShake = true;
                } else if (readAscii.equals("6") && sensorKeyHR && diseaseKey) {
                    bt.send("OK", true);
                    handShake = true;
                }else if (readAscii.equals("7") && sensorKeyBR && diseaseKey) {
                    bt.send("OK", true);
                    handShake = true;
                }else if (readAscii.equals("8") && sensorKeyEC && diseaseKey) {
                    bt.send("OK", true);
                    handShake = true;
                }else {AlertDialog.Builder builder = new AlertDialog.Builder(oneStopService.this);
                    builder.setTitle("Communication Error");
                    builder.setMessage("Restart Scanner and re-connect");

                    // add the buttons
                    builder.setPositiveButton("Reconnect", new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                            // do something ...
                            dialog.dismiss();
                            if (bt.getServiceState() == BluetoothState.STATE_CONNECTED)
                                bt.disconnect();
                            connectScanner.setVisibility(View.VISIBLE);
                            sensorDisplay.setVisibility(View.GONE);
                            mDisplay.setVisibility(View.GONE);
                            collectData.setVisibility(View.GONE);
                            dispResult.setVisibility(View.GONE);
                            shareResult.setVisibility(View.GONE);
                            sensorStatus.setVisibility(View.GONE);
```

```
                        layoutIntro.setVisibility(View.VISIBLE);
                        arr_received.clear();
                        handShake = false;
                        diseaseKey = false;
                        sensorKeyBT = false;
                        Count.cancel();
                        progressDialog.dismiss();
                        failedHandshake=true;

                    }
                });
                builder.setNegativeButton("Cancel", null);

                // create and show the alert dialog
                AlertDialog dialog = builder.create();
                dialog.show();

            }
          }
        }
      }
    }

});

bt.setBluetoothConnectionListener(new BluetoothSPP.BluetoothConnectionListener() {
    public void onDeviceDisconnected() {
        connectionRead.setText("Status : Not connect");
        connectionRead.setBackgroundColor(Color.parseColor("#D3D3D3"));
        menu.clear();
        getMenuInflater().inflate(R.menu.menu_connection, menu);
    }

    public void onDeviceConnectionFailed() {
        connectionRead.setText("Status : Connection failed");
        connectionRead.setBackgroundColor(Color.parseColor("#D3D3D3"));

        AlertDialog.Builder builder = new AlertDialog.Builder(oneStopService.this);
        builder.setTitle("Connection Error");
        builder.setMessage("Retry to connect");

        // add the buttons
        builder.setPositiveButton("Retry", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                // do something ...
                bt.setDeviceTarget(BluetoothState.DEVICE_OTHER);
                Intent intent = new Intent(getApplicationContext(), DeviceList.class);
                startActivityForResult(intent, BluetoothState.REQUEST_CONNECT_DEVICE);
                dialog.dismiss();
            }
        });
        builder.setNegativeButton("Cancel", null);

        // create and show the alert dialog
        AlertDialog dialog = builder.create();
```

```
                dialog.show();
            }

            public void onDeviceConnected(String name, String address) {
                connectionRead.setText("Status : Connected to " + name);
                connectionRead.setBackgroundColor(Color.parseColor("#228B22"));
                menu.clear();
                getMenuInflater().inflate(R.menu.menu_disconnection, menu);
                sensorDisplay.setVisibility(View.VISIBLE);
                layoutIntro.setVisibility(View.GONE);
                layoutProfile.setVisibility(View.VISIBLE);
                collectData.setVisibility(View.VISIBLE);
                connectScanner.setVisibility(View.GONE);

            }
        });



            /*// set corresponding display for selected sensor
            final LinearLayout hrlayout = (LinearLayout) findViewById(R.id.hr_result);
            final LinearLayout spo2layout = (LinearLayout) findViewById(R.id.spo2_result);
            RadioButton rbOxygen = (RadioButton) findViewById(R.id.rb_o2);
            RadioButton rbheartRate = (RadioButton) findViewById(R.id.rb_hr);
            hrlayout.setVisibility(View.VISIBLE);
            spo2layout.setVisibility(View.GONE);
            RadioGroup radioGroup2 = (RadioGroup) findViewById(R.id.rg1);
            radioGroup2.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
                @Override
                public void onCheckedChanged(RadioGroup group, int checkedId) {
                    if (checkedId == R.id.rb_o2) {
                        sensor=2;
                        hrlayout.setVisibility(View.GONE);
                        spo2layout.setVisibility(View.VISIBLE);
                    } else {
                        hrlayout.setVisibility(View.VISIBLE);
                        spo2layout.setVisibility(View.GONE);
                    }
                }
            });*/

// inflate the screen for info and instruction
        final TextView btnOpenPopup = (TextView) findViewById(R.id.info);
        btnOpenPopup.setOnClickListener(new Button.OnClickListener() {

            @Override
            public void onClick(View arg0) {
                LayoutInflater layoutInflater
                        = (LayoutInflater) getBaseContext()
                        .getSystemService(LAYOUT_INFLATER_SERVICE);
                View popupView = layoutInflater.inflate(R.layout.info_oss, null);
                final PopupWindow popupWindow = new PopupWindow(
                        popupView,
                        ViewGroup.LayoutParams.WRAP_CONTENT,
                        ViewGroup.LayoutParams.WRAP_CONTENT);
        Button btnDismiss = (Button) popupView.findViewById(R.id.dismiss);
        btnDismiss.setOnClickListener(new Button.OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                popupWindow.dismiss();
            }
        });

        popupWindow.showAsDropDown(btnOpenPopup, 50, -30);

    }
});

final TextView btnOpenInstruction = (TextView) findViewById(R.id.inst);
btnOpenInstruction.setOnClickListener(new Button.OnClickListener() {

    @Override
    public void onClick(View arg0) {
        LayoutInflater layoutInflater
                = (LayoutInflater) getBaseContext()
                .getSystemService(LAYOUT_INFLATER_SERVICE);
        View popupView = layoutInflater.inflate(R.layout.inst_oss, null);
        final PopupWindow popupWindow = new PopupWindow(
                popupView,
                ViewGroup.LayoutParams.WRAP_CONTENT,
                ViewGroup.LayoutParams.WRAP_CONTENT);

        Button btnDismiss = (Button) popupView.findViewById(R.id.dismiss);
        btnDismiss.setOnClickListener(new Button.OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                popupWindow.dismiss();
            }
        });

        popupWindow.showAsDropDown(btnOpenPopup, 50, -30);

    }
});

/*final TextView btnFluInfo = (Button) findViewById(R.id.buttonFlu);
btnFluInfo.setOnClickListener(new Button.OnClickListener() {
    @Override
    public void onClick(View arg0) {
        LayoutInflater layoutInflater
                = (LayoutInflater) getBaseContext()
                .getSystemService(LAYOUT_INFLATER_SERVICE);
        View popupView = layoutInflater.inflate(R.layout.info_bodytemp, null);
        final PopupWindow popupWindow = new PopupWindow(
                popupView,
                ViewGroup.LayoutParams.WRAP_CONTENT,
                ViewGroup.LayoutParams.WRAP_CONTENT);
        Button btnDismiss = (Button) popupView.findViewById(R.id.dismiss);
        btnDismiss.setOnClickListener(new Button.OnClickListener() {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                popupWindow.dismiss();
            }
        });
        popupWindow.showAsDropDown(btnOpenPopup, 50, -30);
    }
});
final TextView btnSAInfo = (Button) findViewById(R.id.buttonSA);
btnSAInfo.setOnClickListener(new Button.OnClickListener() {
    @Override
    public void onClick(View arg0) {
        LayoutInflater layoutInflater
                = (LayoutInflater) getBaseContext()
                .getSystemService(LAYOUT_INFLATER_SERVICE);
        View popupView = layoutInflater.inflate(R.layout.info_sleepapnea, null);
        final PopupWindow popupWindow = new PopupWindow(
                popupView,
                ViewGroup.LayoutParams.WRAP_CONTENT,
                ViewGroup.LayoutParams.WRAP_CONTENT);
        Button btnDismiss = (Button) popupView.findViewById(R.id.dismiss);
        btnDismiss.setOnClickListener(new Button.OnClickListener() {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                popupWindow.dismiss();
            }
        });
        popupWindow.showAsDropDown(btnOpenPopup, 50, -30);
    }
});
final TextView btnAsthmaInfo = (Button) findViewById(R.id.buttonAsthma);
btnAsthmaInfo.setOnClickListener(new Button.OnClickListener() {
    @Override
    public void onClick(View arg0) {
        LayoutInflater layoutInflater
                = (LayoutInflater) getBaseContext()
                .getSystemService(LAYOUT_INFLATER_SERVICE);
        View popupView = layoutInflater.inflate(R.layout.info_copd, null);
        final PopupWindow popupWindow = new PopupWindow(
                popupView,
                ViewGroup.LayoutParams.WRAP_CONTENT,
                ViewGroup.LayoutParams.WRAP_CONTENT);
        Button btnDismiss = (Button) popupView.findViewById(R.id.dismiss);
        btnDismiss.setOnClickListener(new Button.OnClickListener() {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                popupWindow.dismiss();
            }
        });
        popupWindow.showAsDropDown(btnOpenPopup, 50, -30);
    }
```

```
        });
        final TextView btnARInfo = (Button) findViewById(R.id.buttonAR);
        btnARInfo.setOnClickListener(new Button.OnClickListener() {
            @Override
            public void onClick(View arg0) {
                LayoutInflater layoutInflater
                        = (LayoutInflater) getBaseContext()
                        .getSystemService(LAYOUT_INFLATER_SERVICE);
                View popupView = layoutInflater.inflate(R.layout.info_arrhythmia, null);
                final PopupWindow popupWindow = new PopupWindow(
                        popupView,
                        ViewGroup.LayoutParams.WRAP_CONTENT,
                        ViewGroup.LayoutParams.WRAP_CONTENT);
                Button btnDismiss = (Button) popupView.findViewById(R.id.dismiss);
                btnDismiss.setOnClickListener(new Button.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        // TODO Auto-generated method stub
                        popupWindow.dismiss();
                    }
                });
                popupWindow.showAsDropDown(btnOpenPopup, 50, -30);
            }
        });*/
        // connect scanner by bluetooth

        connectScanner.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                bt.setDeviceTarget(BluetoothState.DEVICE_OTHER);
            /*
                                if(bt.getServiceState() == BluetoothState.STATE_CONNECTED)
                                bt.disconnect();*/
                Intent intent = new Intent(getApplicationContext(), DeviceList.class);
                startActivityForResult(intent, BluetoothState.REQUEST_CONNECT_DEVICE);

            }
        });
    }

  // @Override
/* public void onStart() {
        super.onStart();
        if (!mBoundService.isBluetoothEnabled()) {
            Intent intent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(intent, BluetoothState.REQUEST_ENABLE_BT);
        } else {
            if (!mBoundService.isServiceAvailable()) {
                mBoundService.setupService();
                mBoundService.startService(BluetoothState.DEVICE_ANDROID);
                setup();
            }
        }
    }*/
```

```
public void onDestroy() {
   super.onDestroy();
   bt.stopService();
}

public boolean onCreateOptionsMenu(Menu menu) {
   this.menu = menu;
   getMenuInflater().inflate(R.menu.menu_connection, menu);
   return true;
}

public boolean onOptionsItemSelected(MenuItem item) {
   int id = item.getItemId();
   if (id == R.id.menu_android_connect) {
      bt.setDeviceTarget(BluetoothState.DEVICE_ANDROID);
      /*
                        if(bt.getServiceState() == BluetoothState.STATE_CONNECTED)
                        bt.disconnect();*/
      Intent intent = new Intent(getApplicationContext(), DeviceList.class);
      startActivityForResult(intent, BluetoothState.REQUEST_CONNECT_DEVICE);


   } else if (id == R.id.menu_device_connect) {
      bt.setDeviceTarget(BluetoothState.DEVICE_OTHER);
                        /*
                        if(bt.getServiceState() == BluetoothState.STATE_CONNECTED)
                        bt.disconnect();*/
      Intent intent = new Intent(getApplicationContext(), DeviceList.class);
      startActivityForResult(intent, BluetoothState.REQUEST_CONNECT_DEVICE);
   } else if (id == R.id.menu_disconnect) {
      if (bt.getServiceState() == BluetoothState.STATE_CONNECTED)
         bt.disconnect();
      connectScanner.setVisibility(View.VISIBLE);
      sensorDisplay.setVisibility(View.GONE);
      mDisplay.setVisibility(View.GONE);
      collectData.setVisibility(View.GONE);
      dispResult.setVisibility(View.GONE);
      shareResult.setVisibility(View.GONE);
      sensorStatus.setVisibility(View.GONE);
      layoutIntro.setVisibility(View.VISIBLE);
      arr_received.clear();
      arr_hex.clear();
      handShake = false;
      diseaseKey = false;
      sensorKeyBT = false;
   }

   else if (id == R.id.menu_reinitialize) {
      layout1.setVisibility(View.VISIBLE);
      sensorDisplay.setVisibility(View.VISIBLE);
      mDisplay.setVisibility(View.GONE);
      collectData.setVisibility(View.VISIBLE);
      sensorStatus.setVisibility(View.GONE);
      dispResult.setVisibility(View.GONE);
      shareResult.setVisibility(View.GONE);
      arr_received.clear();
```

```
    handShake = false;
    diseaseKey = false;
    sensorKeyBT = false;
}

else if (id == R.id.action_save) {
    // Save record to database
    //insertTemp();
    // Exit activity
    //finish();
}
// Respond to a click on the "Share to SCC" menu option
else if (id == R.id.action_share){


    if(s.matches("")) {
        Toast.makeText(getApplicationContext(), "Create Profile First ", Toast.LENGTH_LONG).show();

    }else {

        // Go to cloud activity
        Intent shareIntent = new Intent(oneStopService.this, CloudActivity.class);
        //Bundle extras = new Bundle();
        shareIntent.putExtra("DT", "BT");
        shareIntent.putExtra("profile", s);
        shareIntent.putExtra("EOI", eoiValue);
        shareIntent.putExtra("Time", currentDateTime);
        shareIntent.putExtra("Algorithm", "BT1");
        startActivity(shareIntent);
    }

}
else if (id == R.id.action_sms) {
    String messageToSend = "EOI:" + eoiValue;
    String number = "9018340057";

    SmsManager.getDefault().sendTextMessage(number, null, messageToSend, null, null);
    //finish();
}
else if (id == R.id.action_history) {

    // Create a new intent to open the {@link Temperature History}
    Intent temperatureHistoryIntent = new Intent(oneStopService.this, Temp_HistoryActivity.class);

    // Start the new activity
    startActivity(temperatureHistoryIntent);
    //finish();
}

else if (id == R.id.action_algorithm) {
    Intent algIntent = new Intent(oneStopService.this, FluMethods.class);
    startActivity(algIntent);
}
```

```
    return super.onOptionsItemSelected(item);
}
public void onStart() {
    super.onStart();
    if (!bt.isBluetoothEnabled()) {
        Intent intent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(intent, BluetoothState.REQUEST_ENABLE_BT);
    } else {
        if (!bt.isServiceAvailable()) {
            bt.setupService();
            bt.startService(BluetoothState.DEVICE_ANDROID);
            setup();
            summarizedOutput();
            shareWithSCC();
        }
    }

}
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == BluetoothState.REQUEST_CONNECT_DEVICE) {
        if (resultCode == Activity.RESULT_OK)
            bt.connect(data);
    } else if (requestCode == BluetoothState.REQUEST_ENABLE_BT) {
        if (resultCode == Activity.RESULT_OK) {
            bt.setupService();
            bt.startService(BluetoothState.DEVICE_ANDROID);
            setup();
        } else {
            Toast.makeText(getApplicationContext()
                    , "BluetoothActivity was not enabled."
                    , Toast.LENGTH_SHORT).show();
            finish();
        }
    }
}


public void setup() {

    collectData.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            {

                bt.send("OS", true);
                arr_received.clear();
                arr_respiration.clear();
                // progress indicator
                final ProgressDialog progressDialog = new ProgressDialog(oneStopService.this,
                        R.style.AppTheme_Dark_Dialog);
                progressDialog.setIndeterminate(true);
                progressDialog.setMessage("Handshaking & Collecting data...");

                new CountDownTimer(1200, 100) {

                    public void onTick(long millisecondsUntilDone) {
```

```
    progressDialog.show();
}

@Override
public void onFinish() {
    Log.i("Done", "Count Down Timer Finished");
    progressDialog.dismiss();
    if(handShake){
        sensorDisplay.setVisibility(View.GONE);
        collectData.setVisibility(View.GONE);
        // make compute button visible
        dispResult.setVisibility(View.VISIBLE);
        sensorStatus.setVisibility(View.VISIBLE);
        //statusTemp.setBackgroundColor(Color.parseColor("#4CAF50"));
    }else if(failedHandshake){
        // do nothing
    }else{
        AlertDialog.Builder builder = new AlertDialog.Builder(oneStopService.this);
        builder.setTitle("No Communication");
        builder.setMessage("Restart Scanner and re-connect");

        // add the buttons
        builder.setPositiveButton("Reconnect", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                // do something ...
                dialog.dismiss();
                if (bt.getServiceState() == BluetoothState.STATE_CONNECTED)
                    bt.disconnect();
                connectScanner.setVisibility(View.VISIBLE);
                sensorDisplay.setVisibility(View.GONE);
                mDisplay.setVisibility(View.GONE);
                collectData.setVisibility(View.GONE);
                dispResult.setVisibility(View.GONE);
                shareResult.setVisibility(View.GONE);
                sensorStatus.setVisibility(View.GONE);
                layoutIntro.setVisibility(View.VISIBLE);
                arr_received.clear();
                handShake = false;
                diseaseKey = false;
                sensorKeyBT = false;

            }
        });
        builder.setNegativeButton("Cancel", null);

        // create and show the alert dialog
        AlertDialog dialog = builder.create();
        dialog.show();
    }
    if (arr_received.size() > 100) {
        statusTemp.setBackgroundColor(Color.parseColor("#4CAF50"));
        postStatus.setText("2. Click on COMPUTE SEVERITY ");
        dispResult.setClickable(true);
    } else {
        statusTemp.setBackgroundColor(Color.parseColor("#000000"));
```

```
                    postStatus.setText("2. Position Sensor properly, Refresh and collect data again");
                    dispResult.setClickable(false);
                }
                // after timer delay
                    /*sensorDisplay.setVisibility(View.GONE);
                    collectData.setVisibility(View.GONE);
                    // make compute button visible
                    dispResult.setVisibility(View.VISIBLE);
                    sensorStatus.setVisibility(View.VISIBLE);
                    statusTemp.setBackgroundColor(Color.parseColor("#4CAF50"));*/
                    /*AlertDialog alertDialog = new AlertDialog.Builder(oneStopService.this).create();
                    alertDialog.setTitle("Instruction");
                    alertDialog.setMessage("Check sensor status and click on COMPUTE SEVERITY");
                    alertDialog.setButton(AlertDialog.BUTTON_NEUTRAL, "OK",
                            new DialogInterface.OnClickListener() {
                                public void onClick(DialogInterface dialog, int which) {
                                    dialog.dismiss();
                                }
                            });
                    alertDialog.show();*/
            }
        }.start();



        }
    }
  });
}



@Override
public void onStop() {
  super.onStop();

}

public void summarizedOutput() {


  dispResult.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      {

        if (handShake) {

          fluAlgorithm();
          sleepapneaAlgorithm();
          arrhythmiaAlgorithm();
          asthmaAlgorithm();
          if(temperature>100) {
            LayoutInflater layoutInflater
                    = (LayoutInflater) getBaseContext()
```

```
                        .getSystemService(LAYOUT_INFLATER_SERVICE);
                    View popupView = layoutInflater.inflate(R.layout.flu_symp, null);
                    final PopupWindow popupWindow = new PopupWindow(
                        popupView,
                        ViewGroup.LayoutParams.WRAP_CONTENT,
                        ViewGroup.LayoutParams.WRAP_CONTENT);

                    Button btnDismiss = (Button) popupView.findViewById(R.id.dismiss);
                    ch1 = (CheckBox) popupView.findViewById(R.id.checkBox1);
                    ch2 = (CheckBox) popupView.findViewById(R.id.checkBox2);
                    btnDismiss.setOnClickListener(new Button.OnClickListener() {

                      @Override
                      public void onClick(View v) {
                        // TODO Auto-generated method stub
                        popupWindow.dismiss();
                        layout1.setVisibility(View.GONE);
                        layoutIntro.setVisibility(View.GONE);
                        sensorStatus.setVisibility(View.GONE);
                        mDisplay.setVisibility(View.VISIBLE);
                        shareResult.setVisibility(View.VISIBLE);
                        dispResult.setVisibility(View.GONE);
                        fluScreening.setText("Symptomps-Yes");

                      }
                    });

                    popupWindow.showAsDropDown(dispResult, 50, 30);
                  }else {

                    //test.setVisibility(View.GONE);
                    layout1.setVisibility(View.GONE);
                    layoutIntro.setVisibility(View.GONE);
                    sensorStatus.setVisibility(View.GONE);
                    mDisplay.setVisibility(View.VISIBLE);
                    shareResult.setVisibility(View.VISIBLE);
                    dispResult.setVisibility(View.GONE);
                    fluScreening.setText("Symptomps-No");
                  }

                } else {
                  Toast.makeText(getApplicationContext(), "Restart Scanner and Collect Data Again",
Toast.LENGTH_LONG).show();
                }

              }
            }
        });
    }

    // SHARE WITH SCC SERVER

    public void shareWithSCC(){
      //do nothing for now

      shareResult.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            {

                if(s.matches("")) {
                    Toast.makeText(getApplicationContext(), "Create Profile First ", Toast.LENGTH_LONG).show();

                }else{

                    // Go to cloud activity
                    Intent shareIntent = new Intent(oneStopService.this, CloudActivity.class);
                    //Bundle extras = new Bundle();
                    shareIntent.putExtra("DT", "BT");
                    shareIntent.putExtra("profile", s);
                    shareIntent.putExtra("EOI", eoiValue);
                    shareIntent.putExtra("Time", currentDateTime);
                    shareIntent.putExtra("Algorithm", "BT1");
                    startActivity(shareIntent);



                }
            }
        }
    });
}




public void fluAlgorithm()
{
    // do the severity ranking here
    TextView Textv=(TextView) findViewById(R.id.value_flu);
    TextView severityView=(TextView) findViewById(R.id.value_severity_flu);
    ArrayList<Short> arr_trans = new ArrayList<Short>();
    ArrayList<Short> arr_processed1 = new ArrayList<Short>();
    ArrayList<Short> arr_processed2 = new ArrayList<Short>();
    float sum = 0.0f;
    float sum1 = 0.0f;
    float sum2 = 0.0f;
    float avgValue = 0.0f;
    float avgValue1 = 0.0f;
    float avgValue2 = 0.0f;
    float resultVoltage=0.0f;
    double temperature=0.0f;



    // make main display visible and hide arrows
    mDisplay.setVisibility(View.VISIBLE);
    ImageView arrow1=(ImageView) findViewById(R.id.arrow1);
    ImageView arrow2=(ImageView) findViewById(R.id.arrow2);
    ImageView arrow3=(ImageView) findViewById(R.id.arrow3);
    ImageView arrow4=(ImageView) findViewById(R.id.arrow4);
    ImageView arrow5=(ImageView) findViewById(R.id.arrow5);
ImageView arrow6=(ImageView) findViewById(R.id.arrow6);
ImageView arrow7=(ImageView) findViewById(R.id.arrow7);
ImageView arrow8=(ImageView) findViewById(R.id.arrow8);
ImageView arrow9=(ImageView) findViewById(R.id.arrow9);
ImageView arrow10=(ImageView) findViewById(R.id.arrow10);
ImageView arrow11=(ImageView) findViewById(R.id.arrow11);

// hide arrows
arrow1.setVisibility(View.INVISIBLE);
arrow2.setVisibility(View.INVISIBLE);
arrow3.setVisibility(View.INVISIBLE);
arrow4.setVisibility(View.INVISIBLE);
arrow5.setVisibility(View.INVISIBLE);
arrow6.setVisibility(View.INVISIBLE);
arrow7.setVisibility(View.INVISIBLE);
arrow8.setVisibility(View.INVISIBLE);
arrow9.setVisibility(View.INVISIBLE);
arrow10.setVisibility(View.INVISIBLE);
arrow11.setVisibility(View.INVISIBLE);

// display when there is no data
if (arr_received.size() == 0) {
   mDisplay.setVisibility(View.VISIBLE);
   Textv.setText("No Data");
   // Veoi.setText("No Data");
} else {

   // get date and time

   currentDateTime = DateFormat.getDateTimeInstance().format(new Date());
   // initialize the screen
   //Vdatetime.setText("");
   Textv.setText("");
   //Vprompt.setText("");
   //Veoi.setText("");

   // temperature processing begin

  /* for (int i = 0; i < arr_received.size(); i++) {
      if ((arr_received.get(i)>0)&&(arr_received.get(i)<9000)) {
         arr_trans.add(arr_received.get(i));
         Log.i("transferrred", "" + arr_received.get(i));
      }
   }*/


  short value=arr_received.get(0);
   for (int i=0;i<arr_received.size();i++)
   {
      short currentValue=arr_received.get(i);
      value+=((currentValue - value)/10);
      arr_processed1.add(i,value);
      try {
         writeToCsv(Integer.toString(value));
      } catch (IOException e) {
         // TODO Auto-generated catch block
```

```
        e.printStackTrace();
    }
}



// *****Feature 2 ***********************************************************************
// step-1-find maxima

int max =arr_processed1.get(0);
for (int l=0;l<arr_processed1.size(); l++){
    if(arr_processed1.get(l)> max){
        max = arr_processed1.get(l);
    }
}
Log.i("feature21", "" + max);
// step-2-find maxima index

int indexOfMaxima=0;

for (int m=0; m<arr_processed1.size(); m++)

{
    if (max==arr_processed1.get(m)){
        indexOfMaxima=m;
        break;
    }
}

// step-3-find maxima level

float sumMaxima=0;
for (int n=indexOfMaxima-10; n<indexOfMaxima+10; n++)

{
    sumMaxima+=arr_processed1.get(n);
}
float avgMaxima=sumMaxima/20;
Log.i("feature22", "" + avgMaxima);


// ****feature 3************************************************************************

// step-1-transfer to new array

for (int q = 99; q < arr_processed1.size(); q++) {

    arr_trans.add(arr_processed1.get(q));

}

// step-2-find index of delay
int indexOfDelay=0;

for (int p=0; p<arr_trans.size(); p++)
```

```
{
   if (((arr_trans.get(p))-2450)<5){
      indexOfDelay=p;
      break;
   }
}
Log.i("feature3", "" + indexOfDelay);

// *****Feature 1 ******************************************************************

// step-1-find minima

int min =arr_trans.get(0);
for (int i=0;i<arr_trans.size(); i++){
   if(arr_trans.get(i)< min){
      min = arr_trans.get(i);
   }
}


// step-2-find minima index

int indexOfMinima=0;

for (int j=0; j<arr_trans.size(); j++)

{
   if (min==arr_trans.get(j)){
      indexOfMinima=j;
      break;
   }
}
Log.i("feature11", "" + min);

// step-3-find minima level

float sumMinima=0;
for (int k=indexOfMinima; k<indexOfMinima+10; k++)

{
   sumMinima+=arr_trans.get(k);
}
float avgMinima=sumMinima/10;
Log.i("feature12", "" + avgMinima);

// ******Feature 4 ***********

for (int s = 0; s < arr_trans.size(); s++) {
   sum += arr_trans.get(s);
}
avgValue = sum / arr_trans.size();

Log.i("feature4", "" + avgValue);

// ********** Multivariate regression ***********************************************
// equation for temperature
```

```
temperature= 230.0-0.00142*avgMinima-0.04203*avgMaxima-0.21037*indexOfDelay;

double temp2=228.6-0.04243*avgMaxima-0.21267*indexOfDelay;
Log.i("temp", "" + temp2);
Log.i("sizer", "" + arr_received.size());
Log.i("sizet", "" + arr_trans.size());
Log.i("sizep", "" + arr_processed1.size());

try {
   sTemperature = String.valueOf(new DecimalFormat("###.##").format(temperature));
   ratingOfEOI = (temperature - 97) / 10;
   if(ratingOfEOI<0){
      ratingOfEOI=0;
   }else if(ratingOfEOI>1){
      ratingOfEOI=1;
   }
   sEOI = new DecimalFormat("##.##").format(ratingOfEOI);
   sSeverity = new DecimalFormat("##.##").format(100 * ratingOfEOI);

   if (temperature <= 97.5) {
      //prompt = "Normal Temperature";
      arrow1.setVisibility(View.VISIBLE);
      //sEOI="0.0";
      //sSeverity="0.0";
   } else if (temperature <= 98.5) {
      //prompt = "Normal Temperature";
      arrow2.setVisibility(View.VISIBLE);
   } else if (temperature <= 99.5) {
      //prompt = "Normal Temperature";
      arrow3.setVisibility(View.VISIBLE);
   } else if (temperature <= 100.5) {
      //prompt = "Normal Temperature";
      arrow4.setVisibility(View.VISIBLE);
   } else if (temperature <= 101.5) {
      //prompt = "Low Fever,\nconsider consulting your doctor";
      arrow5.setVisibility(View.VISIBLE);
   } else if (temperature <= 102.5) {
      //prompt = "Medium Fever,\nConsult your doctor";
      arrow6.setVisibility(View.VISIBLE);
   } else if (temperature <= 103.5) {
      //prompt = "High Fever,\nConsult your doctor";
      arrow7.setVisibility(View.VISIBLE);
   } else if (temperature <= 104.5) {
      //prompt = "High Fever,\nConsult your doctor";
      arrow8.setVisibility(View.VISIBLE);
   } else if (temperature <= 105.5) {
      //prompt = "Very High Fever,\nConsult your doctor immediately";
      arrow9.setVisibility(View.VISIBLE);
   } else if (temperature <= 106.5) {
      //prompt = "Very High Fever,\nConsult your doctor immediately";
      arrow10.setVisibility(View.VISIBLE);
   } else if (temperature >= 106.5) {
      //prompt = "Extremely High Fever,\nConsult your doctor immediately";
      arrow11.setVisibility(View.VISIBLE);
   }
} catch (NumberFormatException e) {
```

```
            //prompt = "Invalid Data";
            gradientFlu.setVisibility(View.INVISIBLE);
        }



//------ displaying result

        //Vdatetime.setText(currentDateTime);

        //Textv.append(sTemperature+"°F");
        Textv.setText(String.format("%.1f",temperature)+"°F");

        eoiValue=String.format("%.2f",ratingOfEOI);
        severityView.setText(sSeverity);


        //Vprompt.append(prompt );
        //Veoi.append("fluSeverity(100) = " + result);

        //Veoi.append( result);
      }

      // end temperature processing
      arr_received.clear();
      Log.i("sizearr_received", "" + arr_received.size());
      arr_trans.clear();
      arr_processed1.clear();
      arr_processed2.clear();
      fileSeq++;
      timeKeyBT = false;
      diseaseKey = false;
      sensorKeyBT = false;

    };

    public void sleepapneaAlgorithm()
    {
      Log.i("respiration",""+ arr_respiration.size());
      // end temperature processing
      arr_respiration.clear();
      timeKeyBT = false;
      diseaseKey = false;
      sensorKeyBT = false;
    }

    public void arrhythmiaAlgorithm()
    {
      // do the severity ranking here
    }

    public void asthmaAlgorithm()
    {
      // do the severity ranking here
    }
    //write to csv file
    public void writeToCsv(String x) throws IOException {

        Calendar c = Calendar.getInstance();
        File folder = new File(Environment.getExternalStorageDirectory() + "/project");
        boolean success = true;
        if (!folder.exists()) {
            success = folder.mkdir();
        }
        if (success) {
            // Do something on success
            //String fileName = "flu" + String.valueOf(currentDateTime) + ".csv";

            SimpleDateFormat formatter = new SimpleDateFormat("yyyy_MM_dd_HH_mm", Locale.US);
            Date now = new Date();
            String fileName = fileSeq+formatter.format(now)+ ".csv";
            String csv = "/storage/emulated/0/project/"+fileName;
            FileWriter file_writer = new FileWriter(csv, true);
            String s = c.get(Calendar.YEAR) + "," + (c.get(Calendar.MONTH) + 1) + "," + c.get(Calendar.DATE) + ","
+ c.get(Calendar.HOUR) + "," + c.get(Calendar.MINUTE) + "," + c.get(Calendar.SECOND) + "," +
c.get(Calendar.MILLISECOND) + "," + x + "\n";

            file_writer.append(s);
            file_writer.close();


        }
    }
}
```