M. Engg. Project

Real Time Stress Alert for Drivers in Sound Polluted Environment

Submitted by
Md. Ali Hossain
Student ID 0413052009

Supervised by
Dr. Mahmuda Naznin

Submitted to
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
Dhaka 1000, Bangladesh

in partial fulfillment of the requirements for the degree of
Master of Engineering in Computer Science and Engineering

September 2019

Candidate's Declaration

I, do, hereby, certify that the work presented in this project, titled, "Real Time Stress Alert for Drivers in Sound Polluted Environment", is the outcome of the investigation and research carried out by me under the supervision of Dr. Mahmuda Naznin, Professor, Department of CSE, BUET.

I also declare that neither this project nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Md. Ali Hossain

0413052009

The project titled "**Real Time Stress Alert for Drivers in Sound Polluted Environment**", submitted by Md. Ali Hossain, Student ID 0413052009, Session April 2013, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory in partial fulfillment of the requirements for the degree of Master of Engineering in Computer Science and Engineering and approved as to its style and contents on September 15, 2019.

Board of Examiners

1. ____________________

Dr. Mahmuda Naznin — Chairman
Professor — (Supervisor)
Department of CSE, BUET, Dhaka

2. ____________________

Dr. Md. Mostofa Akbar — Member
Professor
Department of CSE, BUET, Dhaka

3. ____________________

Dr. A. B. M. Alim Al Islam — Member
Associate Professor
Department of CSE, BUET, Dhaka

Acknowledgement

I would like to express my deep sense of gratitude to my supervisor, Dr. Mahmuda Naznin, for introducing me the interesting domain of remote health monitoring and teaching me how to perform research works. This project could not be accomplished without her invaluable help and guidance throughout the course. She constantly encouraged me to continue the research. I am grateful to her for giving me support and confidence.

Dhaka
September 15, 2019

Md. Ali Hossain
0413052009

Contents

List of Figures

Abstract

Stress is one of the primary reasons of road accidents while driving. Sound pollution causes critical impact on the physiological condition, it decreases concentration, and increases the stress levels. Development of technologies for recognizing stress is a noteworthy challenge in the field of accident evasion systems. This research presents stress level identification of a driver during sound polluted environment. A model of stress identification has been designed utilizing individual level on drivers physical states, and impacts on the levels of stress. The proposed stress detection model detects stress by periodically collecting heart rate interval data, and skin temperature data through sensors in the smart watch, movement pattern data through location information, and sound levels data through in the smart phone. If Heart Rate Variability value generated from Electrocardiogram signal is less than a threshold value then stress is detected. If the stress level is high, then system will provide a safety alert, which helps drivers for safe driving.

Chapter 1

Introduction

Safe driving requires driver's mental and physical health stability and as well as drivers fitness [4]. Stress decreases driving ability and causes accidents while driving [5][6]. Therefore, stress detection is important to improve drivers' awareness and performance, which is fundamental for road and traffic safety [2]. The significance of drivers stress analysis has been possible with on health sensors [8][10]. Different biological factors, like ECG signal has been approved as a stress measurement metrics [7]. In driving period, mental stress can be computed from Heart Rate Variability (HRV) [1][9].

Sound pollution causes severe impact on the quality of life and health ailments, such as cardiovascular tribulations, high blood pressure, increased levels of diabetes, changes in social behavior. It induces the burdensome tendencies, decreases concentration, increases the stress levels and psychological problems [3][12][13]. In this project, the impact of sound pollution on drivers mental states has been analyzed. We try to put alert if stress level gets higher for a driver. Moreover, when drivers stress levels are high, safety alerts are provided which may help drivers to concentrate on better driving and also to honk less at the other vehicles. It helps in two major ways. One to help for safe driving and to help to reduce sound pollution.

1.1 Stress

Drivers stress is main causal factors behind road accidents. To diminish the number of road accidents, it is important to monitor driver and driving behavior and to give alert to the driver when he or she is in stressful state. In addition, if it were potential to predict unsafe driving behaviors prior to, this is able to contribute to safe driving. As per one report, the amount of car crashes would be reduced by 10-20 percent by monitoring and foreseeing driver and driving behaviors [4]. A dependable and robust drivers' stressful mental state detection system would send an alert to the driver and thus reduce the number of dangerous circumstances on the road.

We already mentioned that, if it were possible to identity risky driving behavior in advance, this would also be useful in preventing road accidents [7]. Thus, it is attractive to design a system to extract the behavorial model of drivers and to find the impact of the noise.

Stress detection systems have been developed to notify driver risk condition which is based on the degree of stress during real driving. In order to identify this physiological condition many methods have been used like eye glance and on-road metrics, but these methods have been criticized as very costly and are usually difficult to obtain [5]. In the alternative hand, physiological signal analysis, particularly using electrocardiogram (ECG) signal, has been valid as a good way to find completely different physiological conditions. ECG signal has been characterized as reliable, accurate and non-invasive indicator. Many researchers have shown that among diverse physiological signals, heart rate variability (HRV) analysis to observe the influence on autonomic nervous system existed in the human body [1][2]. The autonomic nervous system (ANS) is decomposed into sympathetic nervous system (SNS) and parasympathetic nervous system (PSNS) elements. Imbalance between these two systems will be associate indicators of physiological variation mirrored in HRV measurements.

1.2 Sound Pollution

Sound pollution is an unnecessary sound and significant form of energy, which is emitted by a vibrating body and on reaching the ear causes sensation of hearing through nervous system. The sound pollution generally consists of three inter-related elements - the source, receiver and transmission path followed by the sound to reach receiver. This transmission path is typically the atmosphere through that sound is propagated. However, it will embody structural materials of any building containing the receiver. The sound pollution is an unwanted sound that may cause some psychological and physical stress to the living and non-living objects exposed to it.

Sound pollution is considered as one of the major problems of urban communities that has numerous hazardous impacts on the urban environment and may result in a great deal of costs on the society. The increasing ranges of vehicles, musical instruments, tiny scale industries, urbanization and human activities area unit the most sources of sound pollution. *Sound pollution* refers to a sound without agreeable musical quality or as an unwanted or undesired sound. The sound pollution is commonly measured as sound intensity that is determined in terms of the pressure of sound waves on the eardrums, and the scale is logarithmic. Loudness of sound coincides to the degree of sensation depending on the power of sound and sensitivity of ear. The unit of sound intensity activity is Decibel (dB) and every dB rise depicts ten-fold increase in sound intensity. Sound pollution causes environmental pollution and is a cause of human health hazards [12].

1.2.1 Effects of Sound Pollution

Due to sound pollution, the disorders of human, animal and plant bodies are described in the following lines [12]:

Human Efficiency - Regarding the impact of sound pollution on human efficiency there are number of experiments, which shows that the human efficiency decreases with the increase of sound pollution and it increases with the reduction of sound pollution.

Lack of Concentration - For better quality of work there should be concentration, sound pollution causes lack of concentration. Mostly all the offices are on main road and the sound pollution of traffic or the loud speakers of diverse types of horns, divert attention of people working in the offices.

Memory Loss - The effects of excessive sound pollution could be so severe that either there is a permanent loss of memory or a psychiatric disorder.

Fatigue - Because of sound pollution, people cannot concentrate on their work. Thence they spent longer for finishing the work and that they expertise exhaustion. It creates fatigue.

Digestion Problem - The digestion, stomach contractions, flow of saliva and gastric juices all stop proper working due to the high frequency of sound pollution, because the changes are so marked, repeated exposure to astonishing sound pollution should be kept to a minimum.

Blood Pressure Problem - Sound pollution causes certain diseases in human sue to traffic sound pollution such as the headache, high blood pressure and other stresses among the exposed individuals.

Deafness Disaster - The effect of sound pollution on audition is well recognized. Mechanics, locomotive drivers, telephone operators etc., all have their hearing impairment as a result of sound pollution at the place of work. Physicians and psychologists are of the view that sustained exposure to sound level above 80 to 100 dB is risky and thunderous sound pollution causes temporary or permanent deafness.

Hypertension - Relatively low level of sound pollution affects human health adversely and it may cause hypertension, disrupt sleep and or hinder cognitive development in children.

Impact on Animals - Sound pollution damage the nervous system of animals. They lose management of the mind and should become dangerous.

Impact on Plants - Sound pollution causes poor quality of crops even in a pleasant atmosphere.

1.3 Organization

The rest of the project is organized as follows: Chapter 2 presents a brief discussion on related work. Chapter 3 presents system design and implementation. Chapter 4 presents experimental results. Conclusion is given in Chapter 5.

Stress claims lose in health, productivity and financial prospect. We find this as motivational factor behind this study on automatic stress detection and development of effective alert system. Therefore, the objective of this research project are:

1. To detect if there is any relation of sound pollution on drivers physiological and psychological states.
2. To monitor stress levels of drivers in sound polluted environment.
3. To provide real-time stress alert if there occurs any stress sign.

Chapter 2

Related Work

In this chapter, we talk about some projects related to stress and the impact on driving.

2.1 Heart Rate Variability (HRV)

HRV refers to the variations within the beat intervals or correspondingly within the instant HR. Heart Rate Variability (HRV) is additionally acquainted as ”cycle length variability”, ”RR variability” (where R may be a point reminiscent of the height of the QRS complex of the ECG wave; and RR is that the interval between ordered Rs), and ”heart period variability”. It is the physiological phenomenon of variation within the measure between heart beats. The degree of variability within the HR provides information about the functioning of the nervous control on the HR and also the heart’s ability to respond [15]. Heart Rate Variability is measured by the variation within the beat-to-beat interval. Methods accustomed to detect beats include: ECG, blood pressure, ballistocardiograms, and the pulse wave signal derived from a photo-plethysmograph (PPG). ECG is taken into account superior because it provides a clear waveform, which makes it easier to exclude heartbeats not originating within the sinoatrial node. The term ”NN” is employed in place of RR to stress the fact that the processed beats are ”normal” beats. HRV will be evaluated by variety of methods which can be classified in two major types: Time Domain Methods and Frequency Domain Methods.

Time Domain Methods In time domain methods, either the heart rate at any purpose in time or the intervals between ordered normal complexes are determined. In a very continuous ECG record, every QRS complex (Figure 2.2) is detected, and therefore the normal-to-normal (NN) intervals of the instantaneous heart rate is set. Time Domain Methods area unit analysed to calculate variables such as:

- SDNN: The quality deviation of NN intervals. SDNN is commonly calculated over a 24-hour period. SDANN- the quality deviation of the typical NN intervals calculated

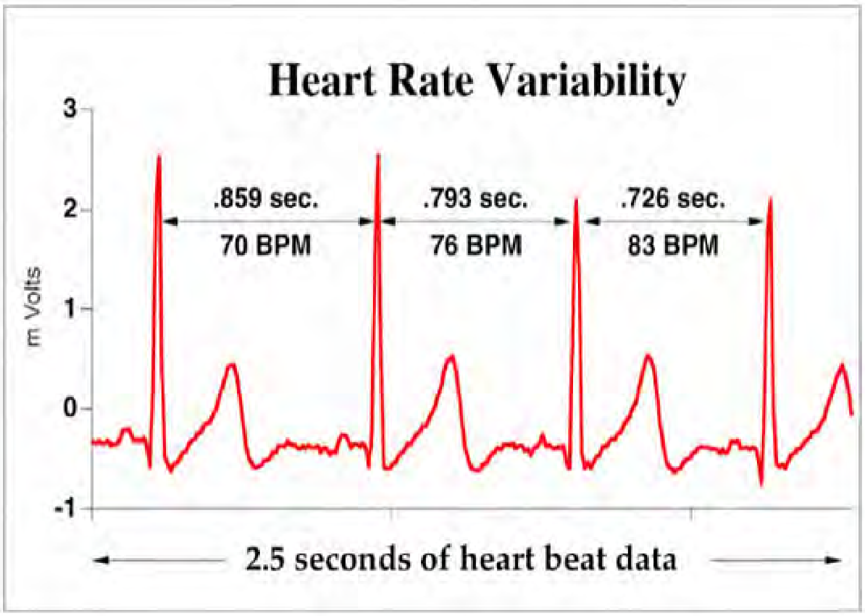


Figure 2.1: Heart Rate Variability [1]

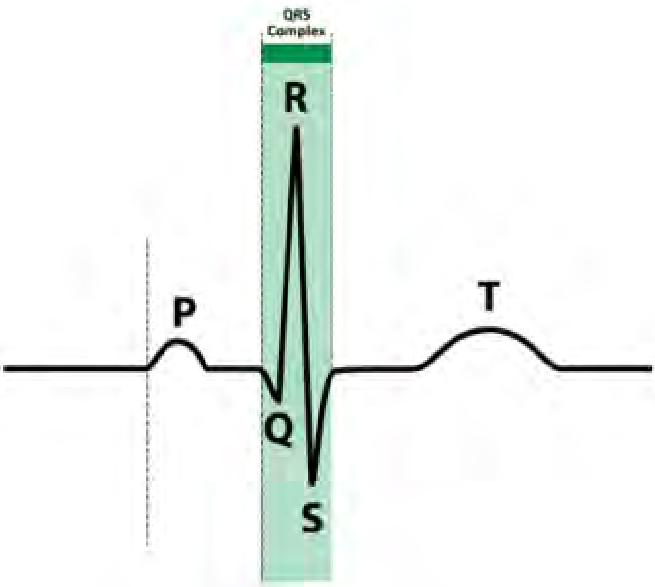


Figure 2.2: QRS Complex [1]

over short periods, typically five minutes. SDANN is thus a measure of changes in heart rate due to cycles longer than five minutes. SDNN reflects all the cyclic parts liable for variability within the period of recording, thus it represents total variability. Equation 2.1 provides the formula to calculate SDNN.

$$SDNN = \sqrt{\frac{1}{N-1}\sum_{n=2}^{N}[I(n) - \bar{I}]^2} \tag{2.1}$$

Here N is that the Total number of heart beats over the time period and $\bar{I}$ is that the mean of RR intervals; calculated as:

$$\bar{I} = \frac{1}{N-1}\sum_{n=2}^{N} I(n) \tag{2.2}$$

- **RMSSD:** RMSSD is the Root mean square of successive differences. RMSSD is that the square root of the mean of the squares of the successive differences between adjacent NNs. It provides estimate of short parts of HRV. RMSSD is set as:

$$RMSSD = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N-1}[I(n) - I(n+1)]^2} \tag{2.3}$$

- **SDSD:** Standard deviation of successive differences. SDSD provides the quality deviation of the serial variations between adjacent NNs.
- **NN50:** The quantity of pairs of serial NNs that dissent by over fifty ms.
- **pNN50:** The proportion of NN50 divided by total number of NNs.
- **NN20:** The quantity of pairs of serial NNs that dissent by over twenty ms.
- **pNN20:** The proportion of NN20 divided by total range of NNs.
- **EBC:** Estimated breath cycle. EBC is usually provided in knowledge acquisition eventualities wherever HRV feedback in real time is a primary goal. EBC is that the range (max-min) within a moving window of a given time duration within the study period. The windows will move in in an exceedingly self-overlapping method or be strictly distinct (sequential) windows. EBC derived from over ten seconds and sixteen seconds ordered and overlapping windows has been shown to correlate extremely with SDNN.

While SDANN and SDNN estimates long term components and overall HRV respectively, for estimation of short term components RMSSD is treated as a useful measure [16].

Frequency Domain Methods

Frequency domain HRV metrics are supported on the calculable power spectral density (PSD) of the NN (normal to normal RR) intervals. These strategies assign bands of frequency and then count the number of NN intervals that match every band. The common definition of bands [17] are as follows:

Total HRV power	**0 - 0.5 Hz**
Ultra-low frequency (ULF) power	0 - 0.0033 Hz
Very low frequency (VLF) power	0.0033 - 0.04 Hz
Low frequency (LF) power	0.04 - 0.15 Hz
High frequency (HF) power	0.15 - 0.4 Hz
Very high frequency (VHF) power	0.4 - 0.5 Hz

In addition, the LF/HF quantitative relation is usually cited as a parameter of interest. In healthy adults, the LF/HF quantitative relation is often between 1.5 and 4.5. Several methods of frequency domain analysis are in follow. Power spectral density (PSD), using parametric or non-parametric methods, provides basic information on the power distribution across frequencies. one in every of the foremost ordinarily used PSD methods is that the discrete Fourier transform. Methods for the calculation of PSD could also be usually classified as non-parametric and parametric. In most instances, each methods provide comparable results.

The convenience of the non-parametric methods are:

1. In in-depth cases Fast Fourier Transform [FFT] is customized. The simplicity of the algorithm is one of the advantages of this methodology.
2. The high transform speed.

On the opposite hand the benefits of parametric methods are:

1. Smoother spectral components that can be distinguished independent of pre-selected frequency bands.
2. Easy post process of the spectrum with associate degree calculation of low and high-frequency power parts with a simple identification of the central frequency of of every element.
3. Accurate estimation of PSD even on a tiny low variety of samples on that the signal is meant to keep up stationery.

The basic disadvantage of parametric methods is that the want of verification of the quality of the chosen model and of its complexion(that is, the order of the model). Along with classical FFT-based methods used for the calculation of frequency parameters, a additional appropriate PSD estimation technique is that the Lomb–Scargle (LS) periodogram [17]. Analysis has shown that the LS periodogram can turn out a extra correct estimate of the PSD than FFT methods for typical RR data. Since the RR knowledge is an inconsistently sampled knowledge, another advantage of the LS technique is that in distinction to FFT-based methods it is ready to be used without the ought to re-sample and deterrent the RR data [18].

2.2 Stress Detection

Several studies indicate that HRV can be a useful indicator to understand psychological state of drivers. In our study we have considered, [19] as base values to detect stress. The HRV is measured exploitation sensing element data provided by smart watch. As a sample we have used Microsoft Band 2 [11] which reads Heart Rate, Heart Rate Interval, Skin Temperature as long as the driver is wearing the band. To receive the R-R Interval, an Android application has been developed. Initially after informing the drivers about some basic information figure 4.1 the application asks to provide a few demographic information of the drivers, i.e. name, age, vehicle type, gender etc. The UI of the profile creation is illustrated in figure 4.1. Once the profile is made, the user is asked for Heart Rate Interval subscription consent by the appliance. Once the user grants permission, the application continues reading R-R interval data in background. RMSSD is calculated over 20 RR interval readings, and this gives one stress reading according to the experimental result of Agelink et al. [19]. After taking such readings, if the drivers found any stress sign, an alert will be populated.

This application automatically detects stress without any active participation of the drivers. Privacy of the drivers personal information is fully protected, as the system does not upload any data to any remote server. Therefore it does not require internet connection either. Such automated systems can be of use in monitoring patients with stress who are forgetful of their personal well-being. There is a provision to store stress history in the application database, which can be helpful for the drivers state. Again, this history are often viewed solely with consent of the account possessor. At present the system considers only HRV as indicator of stress and detects stress according to reference RMSSD value of HRV. We have future plan to add more indicators according to our study outcome, and we also plan to make the system context adaptive incorporating machine learning methodologies.

2.3 Background Study

Identification of stress depends on heart rate variability (HRV) examination which is gotten from ECG signal, and reflects autonomic nervous system condition of the human body. The change of autonomic nervous system predicts the feeling of anxiety of drivers during driving activity and permits a safe driving by the probability of an early cautioning. This stress, occurring during driving, is caused by assorted factors, for example, evolving disposition, bio rhythm, weariness, fatigue or ailment which can keep the driver from arriving at unseemly state for driving. The ECG sign of the driver is extracted and preprocessed so as to play out the HRV examination. This investigation is accomplished utilizing one of the domain investigation approach, for example, time, frequency, time-frequency or non-linear methods including Wavelet and STFT. After HRV examination, a few parameters are separated to fabricate a vector of highlights for the order stage. Authors experimentation is performed with information issued from 16 unique subjects from the Stress Recognition in Automobile Driver database (DRIVEDB). A few order systems were examined including support vector machine with radial basis function (SVM-RBF) bit, K nearest neighbor (KNN), and radial basis function (RBF) classifiers. Authors results show that pressure identification could be anticipated with a precision of 83 percent utilizing SVM-RBF classifier. This likewise calls attention to the power of ECG biometric as an exact physiological pointer of a driver state. Figure 2.3 shows the sequence of applied methods of the system [2].

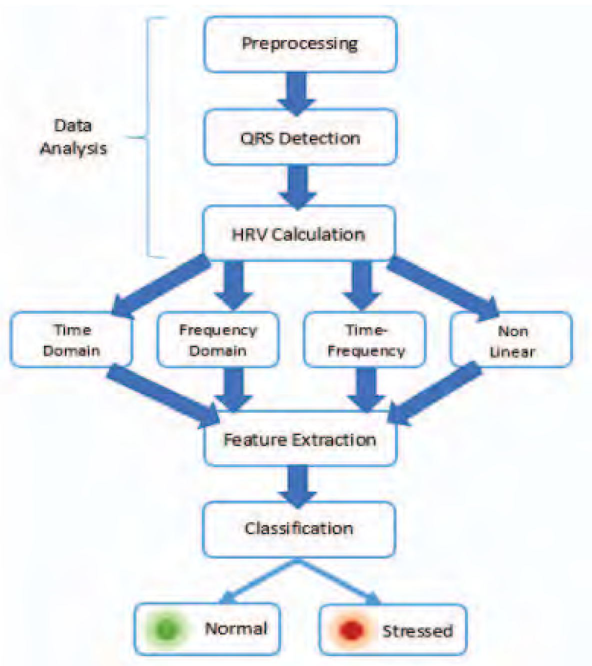


Figure 2.3: Sequence of applied methods [2]

Evaluating 10 to 30 percent of street fatalities are identified with drowsy driving or driver fatigue. Drivers drowsiness identification dependent on natural and vehicle sign is being considered in preventive vehicle wellbeing. Autonomous Nervous System (ANS) action, which can be estimated non-intrusively from the Heart Rate Variability (HRV) signal acquired from surface ECG, presents changes during pressure, outrageous weakness and drowsiness scenes. Authors speculation is that these modifications show on HRV. Authors build up an on-line locator of drivers drowsiness in view of HRV investigation. Two databases have been investigated: one of driving recreation in which subjects were restless, and the other of genuine circumstance with no lack of sleep. An outer onlooker commented on every moment of the accounts as sluggish or wakeful, and establishes creators reference. The proposed indicator arranged sluggish minutes with an affect ability of 0.85 what's more, a prescient positive estimation of 0.93, utilizing 25 highlights. Figure 2.4 shows the components of the system [5].

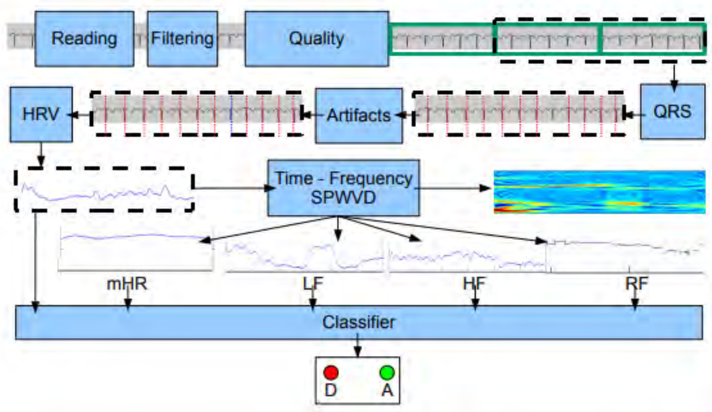


Figure 2.4: Components of the system [5]

Presenting techniques by authors for gathering and analyzing physiological information during genuine world driving undertakings is to decide a drivers relative feeling of anxiety. Electrocardiogram, electromyogram, skin conductance and breath were recorded persistently while drivers pursued a set course through open streets in the more noteworthy Boston region. Information from twenty four drives of in any event fifty minute term were gathered for examination. The information were broke down in two different ways. Investigation one utilized highlights from five moment intervals of information during the rest, parkway and city driving conditions to recognize three degrees of driver worry with an exactness of more than 97 percent over various drivers and driving days. Examination two thought about constant highlights, determined at one second intervals all through the whole drive, with a measurement of noticeable stressors made by autonomous coders from video tapes. The outcomes demonstrate that for most drivers considered, skin conductivity and pulse measurements are most firmly corresponded with driver anxiety. These discoveries demonstrate that physiological sign can give a measurement of driver stress in future vehicles equipped for physiological checking. Such a measurement could be utilized to help oversee non-basic in-vehicle data frameworks and could likewise give a nonstop proportion of how unique street and traffic conditions influence drivers. Figure 2.5 shows the process of the system [6].

Drowsiness of Drivers is one of the fundamental causes of traffic collisions. Driver fatigue is a noteworthy factor in countless vehicle collisions. The advancement of innovations for identifying or averting drowsiness in the drivers seat is a noteworthy test in the field of accident avoidance systems. Because of the peril that drowsiness displays out and about, techniques should be created for checking its effects. Authors portray an ongoing non-meddlesome strategy for distinguishing drowsiness of driver. It utilizes webcam to procure video pictures of the driver. Visual highlights like mouth and eyes which are ordinarily describing the languor of the driver are extracted with the assistance of picture preparing procedures to recognize drowsiness. Figure 2.6 shows the block diagram of the proposed system [21].

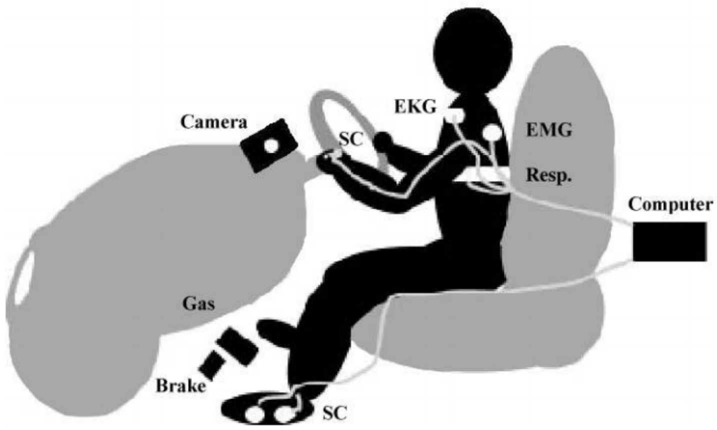


Figure 2.5: Process of the system [6]

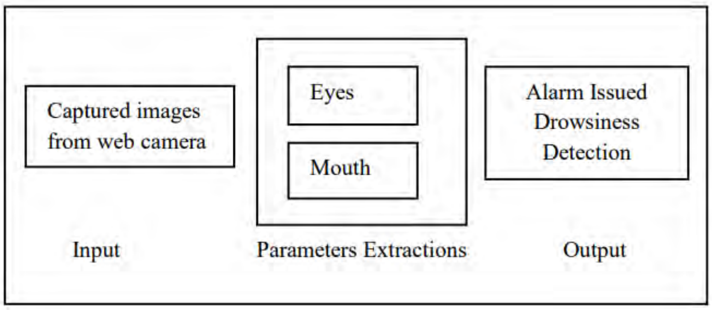


Figure 2.6: Block diagram of the proposed system [21]

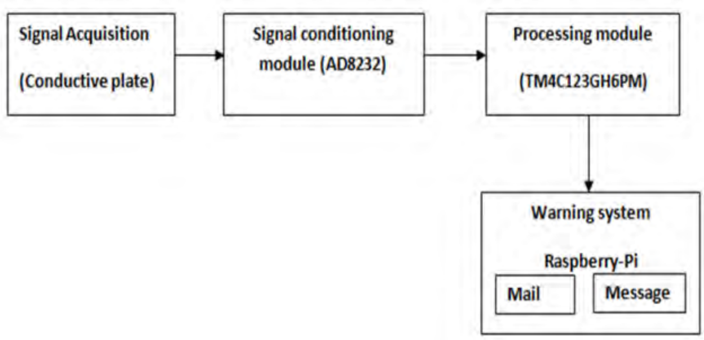


Figure 2.7: System architecture of the proposed system [7]

Authors bargain about the structuring and advancement of a driver alert system dependent on EKG signals to recognize the driver drowsiness and in this manner to lessen accident rate. Conductive sensor connected to the hand which constantly screen drivers pulse and disperse cautioning message through mail and SMS to the proprietor or particular expert. The structure framework contains four parts including signal acquisition unit, signal conditioning unit processing unit and Message delivery system. The sign obtained from the conductive plate contains commotions. Evacuation of the clamor signals from the ECG sign is significant and basic. One of the original thoughts in the work is the advancement of conductive sensor utilizing conductive plates and incorporated sign molding module AD8232 to quantify the pulse. Utilization of a coordinated sign molding module makes the framework little and practical. Authors utilize a Tiva C series micro-controller for processing unit and Raspberry-pi as message delivery system. Mail and message sending highlights of the framework causes the particular specialist to take quick activities. Improvement can be done to the framework by including the GPS usefulness for tracking the area of vehicle. Figure 2.7 shows the system architecture of the proposed system [7].

Chapter 3

System Design

In this chapter, we provide the details of our designed system and the steps of implementation.

3.1 System Architecture

Here, we provide the details of our designed system architecture.

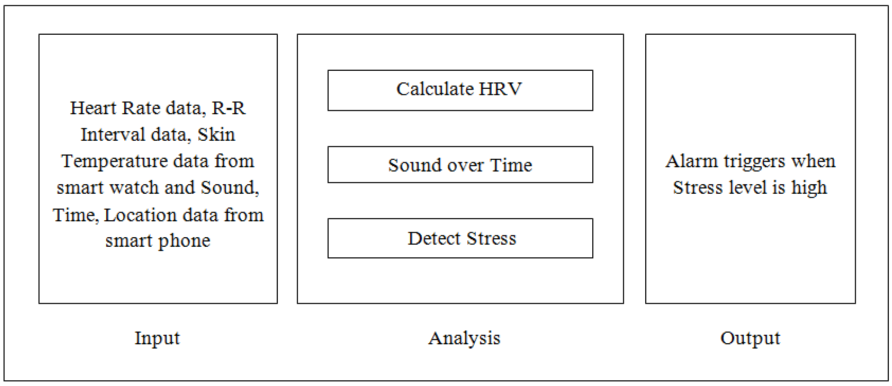


Figure 3.1: Proposed system architecture

The model in the above figure is divided in to three parts, one data is collected from different sensors, data is processed through an android application, and the alert output is generated to alert the driver. We provide the detail design as follows.

3.2 Design

In this section, we provide the detail design. It is clear that, psychological condition of an individual is cogitated in several day-to-day manner and activities. We have planned to design a real time stress alert for drivers in sound polluted environment involving three parts such as 1. Data Collection, 2. Data Analysis, and 3. Providing Alerts.

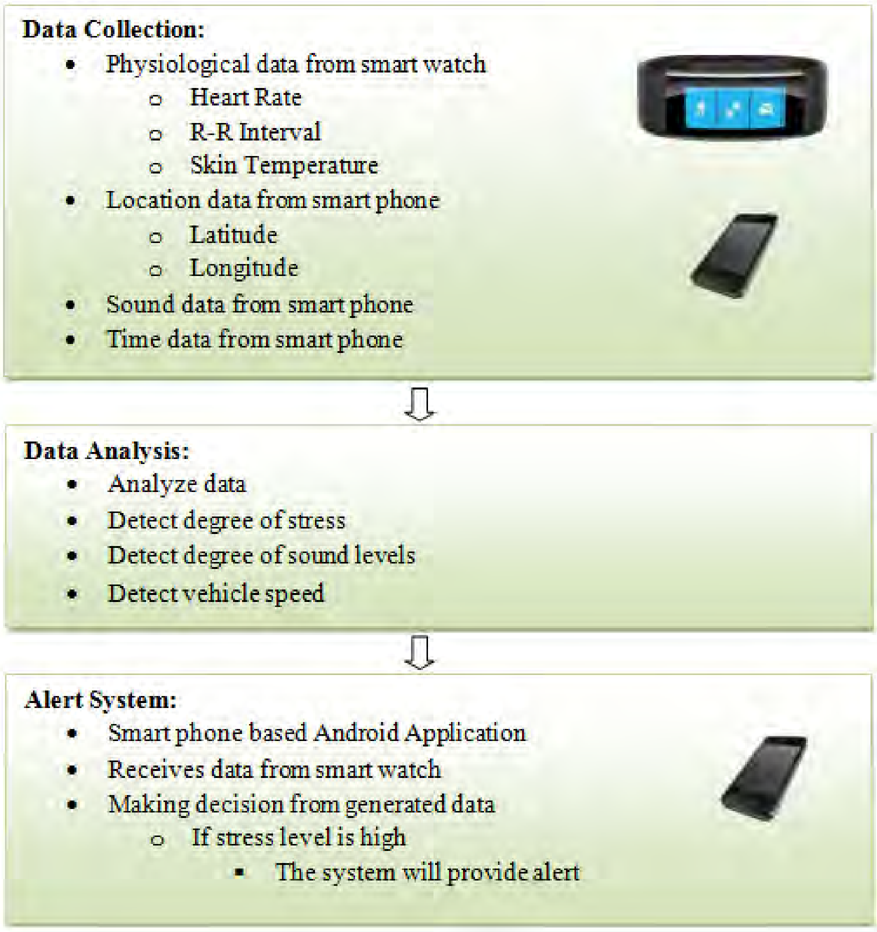


Figure 3.2: Implementation steps.

In Figure 3.2, we find that the system collects data of drivers from the wearable devices. Android application processes the collected data and provide messages.

3.2.1 Part 1: Data Collection

In our system, data will be gathered from smart wearable device and smart phone. The smart wearable device recurrently collects drivers physiological data. Smart wearable device and smart phone are provided with these data on the following:

- **Heart Rate Data:** A study guided by Harvard Medical School [14] shows that, cardiovascular system is directly influenced by mind and mood. Psychological states like anxiety, depression etc. make a condition of emergency readiness, which results in hormone levels rise, blood vessels constrict, and heartbeat speed up. If an individual is seriously anxious, the emergency response becomes constant. Eventually, it damages the blood vessels and makes the heart less sensitive to signals telling it to slow down or speed up as the body's demands change. Our proposed system will monitor these deviations and try to detect stress from the heart rate data [1].

- **R-R Interval Data** The ECG of healthy individuals inspect periodic variation in R-R Intervals the beat to beat variation in either heart rate or the duration of the R-R Interval. QRS detection was achieved with the optimized Pan and Tompkins method explained, and R-R Intervals were obtained.

- **Skin Temperature Data** Our system is designed to track the skin temperature data measured in degree Celsius.

- **Latitude Data** The angular distance of a place North or South of the earth's equator. Our system is designed to track location data with a view to identify the change the vehicle speed. Our study data are collected from latitude 23.902225 to latitude 23.922030. Figure: 3.3 shows the data collection route map.

- **Longitude Data** The angular distance of a place East or West of the earth's equator. Our system is designed to track location with a view to identify the change the vehicle speed. Our study data are collected from longitude 90.412807 to longitude 90.433661. Figure: 3.3 shows the data collection route map.

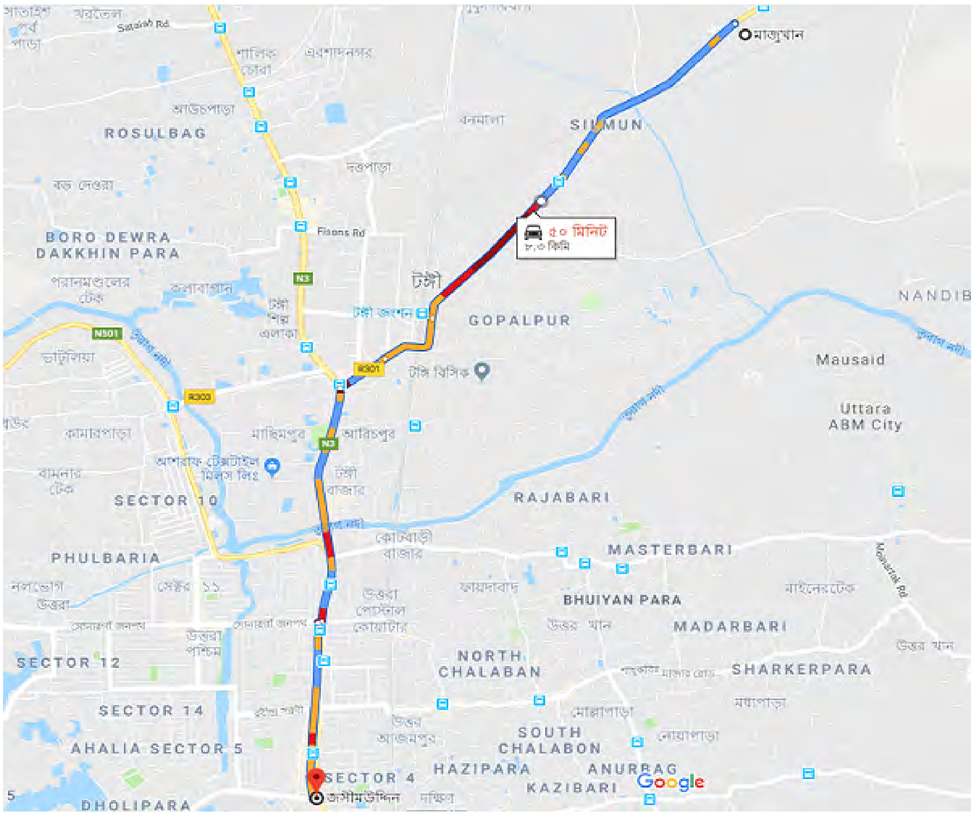


Figure 3.3: Location data collection route map

- **Speed Data** Our system is designed to calculate vehicle speed from location data measured using kilometers per hour (km/h) using smart phone.

- **Sound Data** Our system is designed to keep record sound levels data measured using decibel (dB) using smart phone.
- **Time Data** Our system is designed to keep record times data using smart phone.

3.2.2 Part 2: Data Analysis

Smart wearable device periodically collects data on physical parameters, these data will be sent to the synchronized smart phone, where it will be analyzed to identify specific patterns indicating stress. Once the system detects stress, it triggers the alert system. In the same time, the system keeps record of stress to identify the severity of stress and initiates adaptive alert mechanisms. In the same time, the system keeps the record of the sound levels in Decibel (dB), location data point of latitude and longitude, it calculates vehicle speed from location over time. All of analysis working procedure query are executed in Microsoft SQL Server.

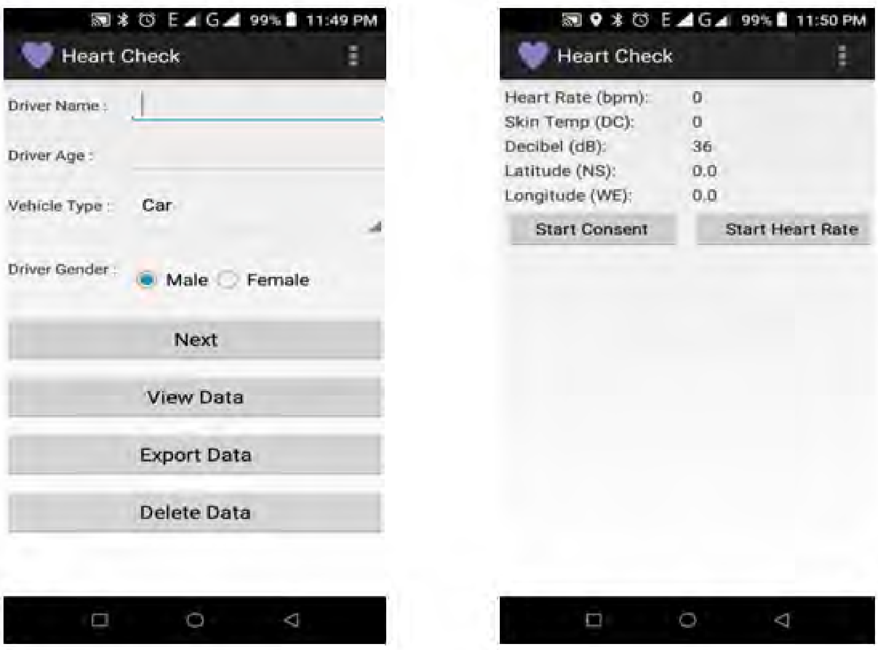
3.2.3 Part 3: Alert System

The alert system might need rigorous calibration. An Android application is designed to act according to the level of stress. If stress level is high then system will provide safety alerts. If stress level is high and speed is high then system will provide safety alerts. If stress level is high and the speed is high and sound level is high then system provides safety alerts.

Chapter 4

Experimental Results

In this chapter, we record data, we analyze and report the results what we have found. In the following figure our alert system is shown.



(a) Driver profile

(b) Physiological data

Figure 4.1: Stress alert application screen shots

To evaluate correctness of the system we have designed an experimental setup. We conducted our research experiment of the system on 18 drivers and at the same time have assessed their psychological condition with an established scale [20]. Finally we have compared outcome of drivers stress status and our system status to measure accuracy of our system.

Drivers: We collected data from 18 drivers and took their consent to conduct the experiments. We approached all of them in person and assured them that the experiment will be completely anonymous and none of their personal information will be shared with anyone. All of them agreed and co-operated. Figure 4.2 shows the age range of the drivers is 28-48 and all of them are male. All of the drivers were sound both mentally and physically and were not consuming any anti-depressant medication.

					Location				Time	
SL	Name	Vehicle Type	Age	Gender	From Latitude	From Longitude	To Latitude	To Longitude	From	To
1	Driver 1	Bus (AC)	36	Male	23.900766	90.411121	23.917483	90.429974	6/24/19 7:42 PM	6/24/19 8:01 PM
2	Driver 2	Leguna	37	Male	23.902225	90.412807	23.868478	90.403312	6/17/19 8:57 AM	6/17/19 9:21 AM
3	Driver 3	Bus	39	Male	23.901583	90.410153	23.921083	90.430154	6/25/19 9:08 AM	6/25/19 9:22 AM
4	Driver 4	Bike	32	Male	23.869297	90.400368	23.871526	90.401591	6/17/19 7:28 PM	6/17/19 7:40 PM
5	Driver 5	Bus	48	Male	23.906602	90.414853	23.916802	90.416854	6/18/19 8:42 AM	6/18/19 8:57 AM
6	Driver 6	Leguna	43	Male	23.922381	90.435189	23.906602	90.414853	6/18/19 8:23 AM	6/18/19 8:37 AM
7	Driver 7	Bus	33	Male	23.900766	90.411121	23.917483	90.429974	6/18/19 7:42 PM	6/18/19 8:01 PM
8	Driver 8	Bus	32	Male	23.897893	90.409205	23.922030	90.433661	6/25/19 7:34 PM	6/25/19 8:00 PM
9	Driver 9	Car	40	Male	23.913225	90.412807	23.869479	90.400312	6/23/19 8:57 AM	6/23/19 9:21 AM
10	Driver 10	Truck	39	Male	23.871526	90.400591	23.922450	90.435032	6/17/19 7:55 PM	6/17/19 8:23 PM
11	Driver 11	Truck	38	Male	23.922381	90.435189	23.906602	90.414957	6/24/19 8:23 AM	6/24/19 8:37 AM
12	Driver 12	Bus	33	Male	23.874251	90.400337	23.874764	90.400676	6/25/19 6:58 PM	6/25/19 7:07 PM
13	Driver 13	Car (AC)	35	Male	23.891536	90.400591	23.942450	90.436732	6/23/19 7:55 PM	6/23/19 8:23 PM
14	Driver 14	Bus	44	Male	23.863876	90.401109	23.864879	90.421106	6/18/19 7:20 PM	6/18/19 7:30 PM
15	Driver 15	Bike	28	Male	23.919649	90.432590	23.901083	90.410153	6/25/19 8:46 AM	6/25/19 9:00 AM
16	Driver 16	Bus	40	Male	23.906602	90.414853	23.926612	90.434873	6/24/19 8:42 AM	6/24/19 8:57 AM
17	Driver 17	Bus	47	Male	23.873872	90.401109	23.883877	90.411189	6/24/19 7:20 PM	6/24/19 7:30 PM
18	Driver 18	Bus	35	Male	23.869297	90.400368	23.877546	90.425593	6/23/19 7:28 PM	6/23/19 7:40 PM

Figure 4.2: Drivers profile data

Experiment setup: The experiment was run on each drivers for at least 30 minutes. For these 30 minutes the put on the smart-watch which is synchronized with an android phone. The smart-watch collected *Heart Rate Interval* of the drivers and sent the data to the smart phone via Bluetooth. The HRV was calculated taking 20 consecutive *Heart Rate Interval* readings. We considered 20 consecutive HRV values to determine psychological state of the drivers without relaxing quality of performance of the system. If the system detects in the range below the standard value according to [19] drivers will be considered stressful by the system.

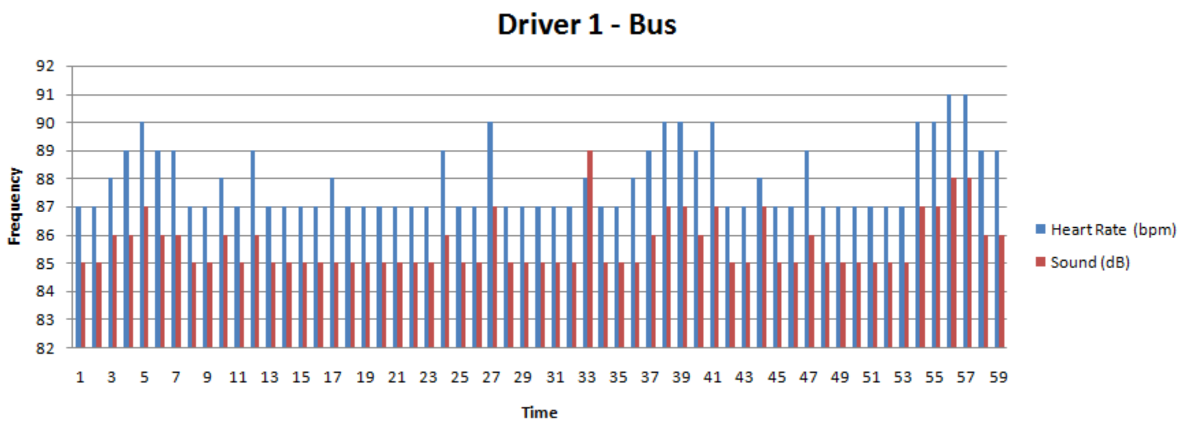


Figure 4.3: AC bus driver 1 - heart rate vs sound level

Figure 4.3 shows the ac bus driver 1 started his journey from location 23.900766, 90.411121 to

23.917483, 90.429974 and time from 6/24/2019 7:42:23 pm to 6/24/2019 8:01:00 pm. Here, we measured the time in seconds (s), heart rate in beats per minutes (bpm) and sound level in decibel (dB). Here, sound range was 85 to 89 db and heart rate range was 87 to 91 bpm. We examined, when sound level is increased then heart rate also increased.

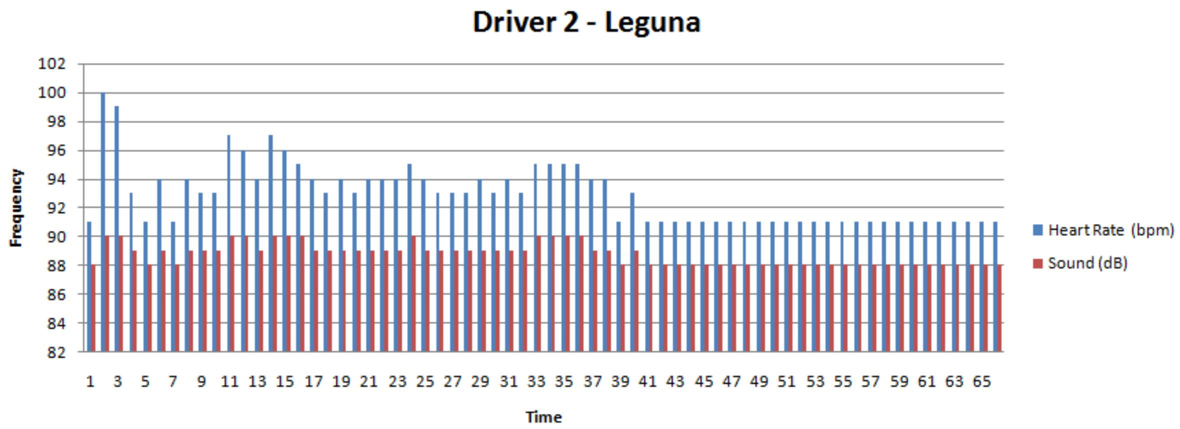


Figure 4.4: Leguna driver 2 - heart rate vs sound level

Figure 4.4 shows the leguna driver 2 started his journey from location 23.902225, 90.412807 to 23.868478, 90.403312 and time from 6/17/2019 8:57:02 am to 6/17/2019 9:21:05 am. Here, we measured the time in seconds (s), heart rate in beats per minutes (bpm) and sound level in decibel (dB). Here, sound range was 88 to 90 db and heart rate range was 91 to 100 bpm. We examined, when sound level is increased then heart rate also increased.

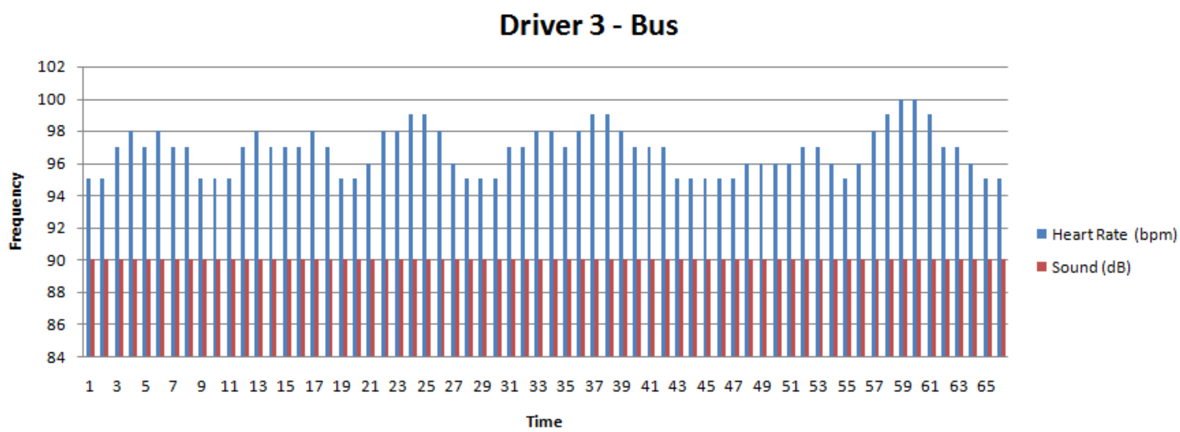


Figure 4.5: Bus driver 3 - heart rate vs sound level

Figure 4.5 shows the bus driver 3 started his journey from location 23.901583, 90.410153 to 23.921083, 90.430154 and time from 6/25/2019 9:08:15 am to 6/25/2019 9:22:30 am. Here, we measured the time in seconds (s), heart rate in beats per minutes (bpm) and sound level in decibel (dB). Here, sound range was 90 to 90 db and heart rate range was 95 to 100 bpm. We examined, sound level is fixed and heart rate is up and down.

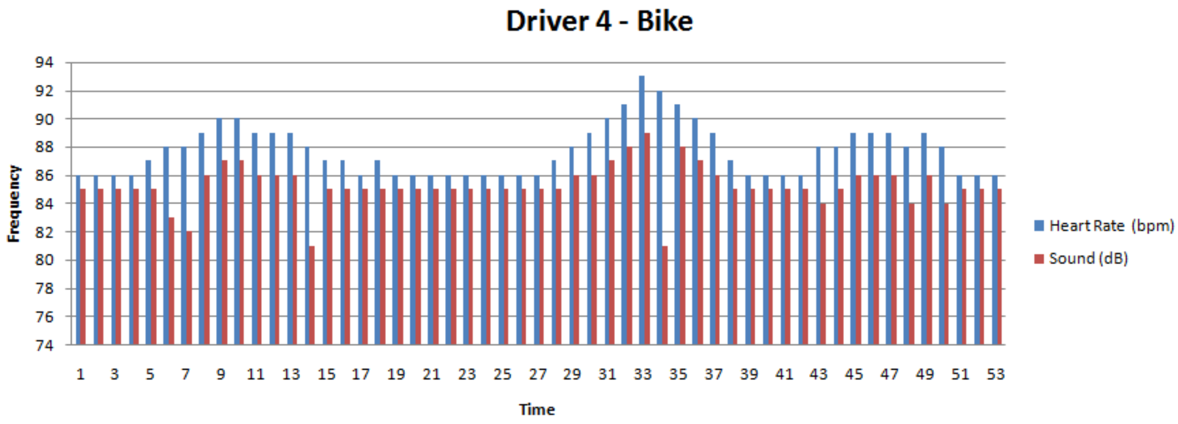


Figure 4.6: Bike driver 4 - heart rate vs sound level

Figure 4.6 shows the bike driver 4 started his journey from location 23.869297, 90.400368 to 23.871526, 90.401591 and time from 6/17/2019 7:28:36 pm to 6/17/2019 7:40:32 pm. Here, we measured the time in seconds (s), heart rate in beats per minutes (bpm) and sound level in decibel (dB). Here, sound range was 81 to 89 db and heart rate range was 86 to 93 bpm. We examined, when sound level is increased then heart rate also jumped.

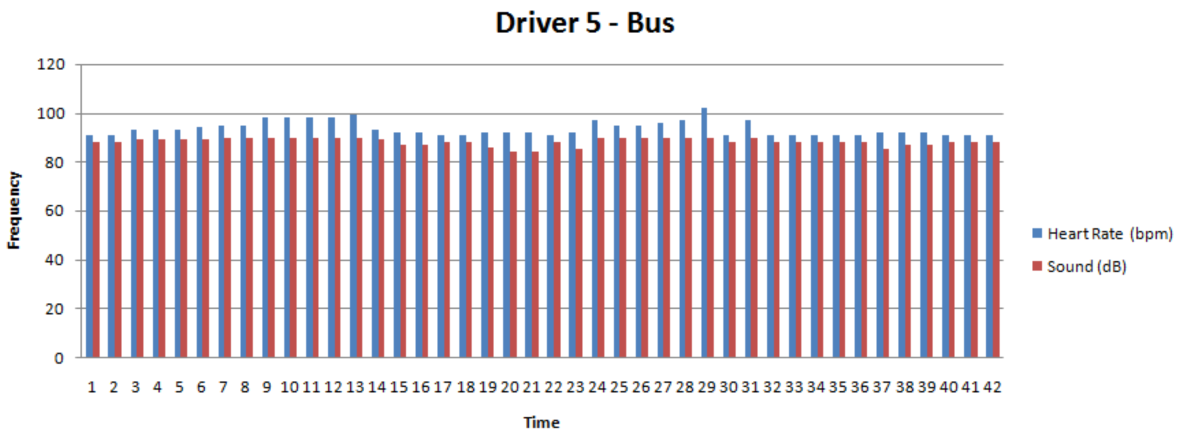


Figure 4.7: Bus driver 5 - heart rate vs sound level

Figure 4.7 shows the bus driver 5 started his journey from location 23.906602, 90.414853 to 23.916802, 90.416854 and time from 6/18/2019 8:42:34 am to 6/18/2019 8:57:21 am. Here, we measured the time in seconds (s), heart rate in beats per minutes (bpm) and sound level in decibel (dB). Here, sound range was 84 to 90 db and heart rate range was 91 to 102 bpm. We examined, when sound level is increased then heart rate also increased.

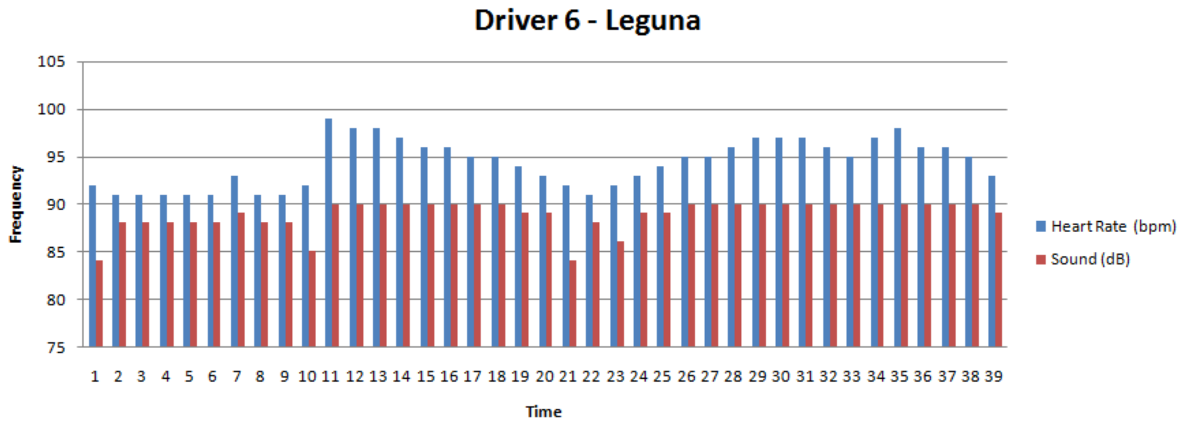


Figure 4.8: Leguna driver 6 - heart rate vs sound level

Figure 4.8 shows the leguna driver 6 started his journey from location 23.922381, 90.435189 to 23.906602, 90.414853 and time from 6/18/2019 8:23:47 am to 6/18/2019 8:37:18 am. Here, we measured the time in seconds (s), heart rate in beats per minutes (bpm) and sound level in decibel (dB). Here, sound range was 84 to 90 db and heart rate range was 91 to 99 bpm. We examined, when sound level is increased then heart rate also increased.

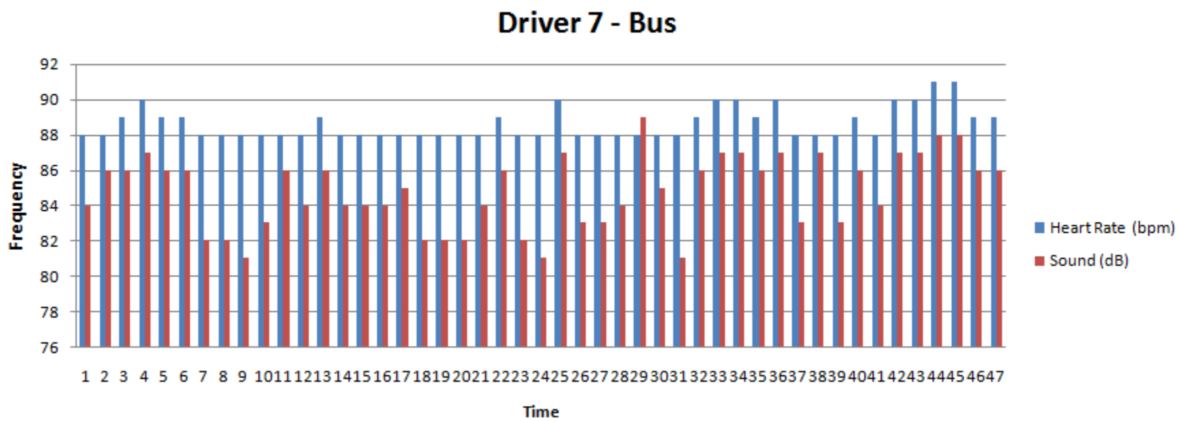


Figure 4.9: Bus driver 7 - heart rate vs sound level

Figure 4.9 shows the bus driver 7 started his journey from location 23.900766, 90.411121 to 23.917483, 90.429974 and time from 6/18/2019 7:42:23 pm to 6/18/2019 8:01:00 pm. Here, we measured the time in seconds (s), heart rate in beats per minutes (bpm) and sound level in decibel (dB). Here, sound range was 81 to 89 db and heart rate range was 88 to 91 bpm. We examined, when sound level is increased then heart rate also increased.

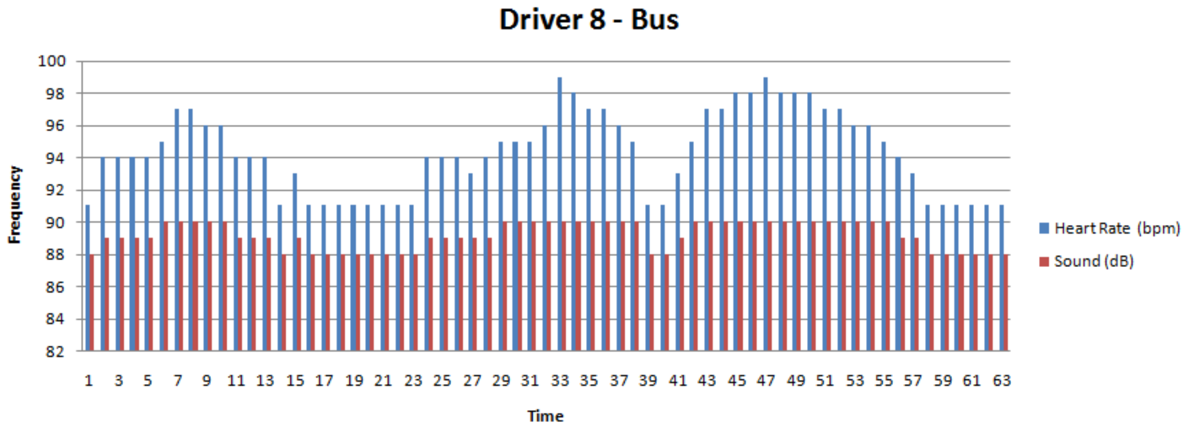


Figure 4.10: Bus driver 8 - heart rate vs sound level

Figure 4.10 shows the bus driver 8 started his journey from location 23.897893, 90.409205 to 23.922030, 90.433661 and time from 6/25/2019 7:34:37 pm to 6/25/2019 8:00:35 pm. Here, we measured the time in seconds (s), heart rate in beats per minutes (bpm) and sound level in decibel (dB). Here, sound range was 88 to 90 db and heart rate range was 91 to 99 bpm. We examined, when sound level is increased then heart rate also increased.

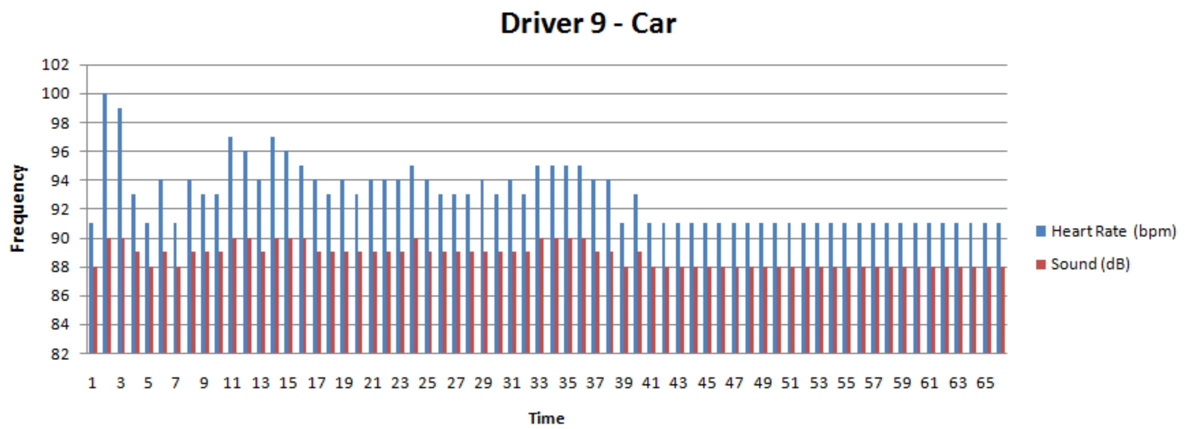


Figure 4.11: Car driver 9 - heart rate vs sound level

Figure 4.11 shows the car driver 9 started his journey from location 23.913225, 90.412807 to 23.869479, 90.400312 and time from 6/23/2019 8:57:02 am to 6/23/2019 9:21:05 am. Here, we measured the time in seconds (s), heart rate in beats per minutes (bpm) and sound level in decibel (dB). Here, sound range was 88 to 90 db and heart rate range was 91 to 100 bpm. We examined, when sound level is increased then heart rate also increased.

Figure 4.12 shows the truck driver 10 started his journey from location 23.871526, 90.400591 to 23.92245, 90.435032 and time from 6/17/2019 7:55:35 pm to 6/17/2019 8:23:10 pm. Here,

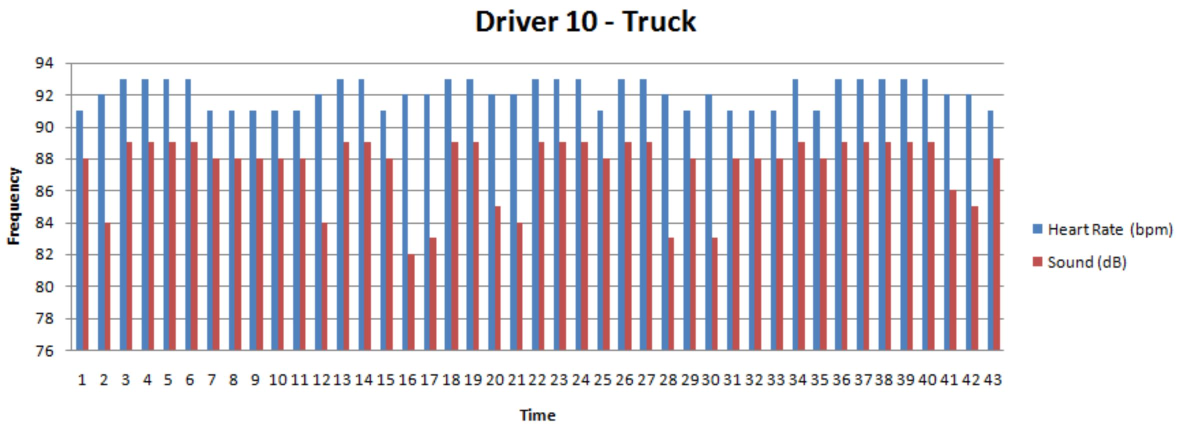


Figure 4.12: Truck driver 10 - heart rate vs sound level

we measured the time in seconds (s), heart rate in beats per minutes (bpm) and sound level in decibel (dB). Here, sound range was 82 to 89 db and heart rate range was 91 to 93 bpm. We examined, when sound level is increased then heart rate also increased.

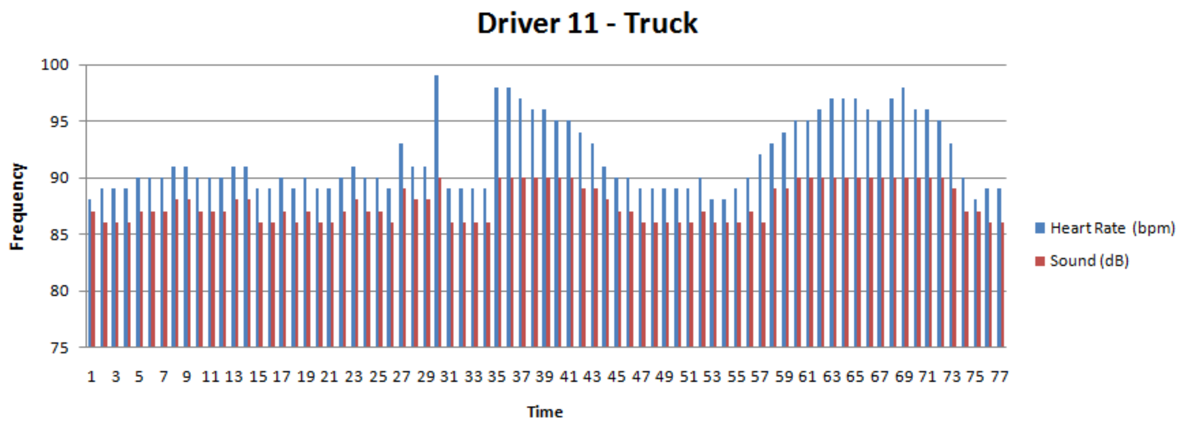


Figure 4.13: Truck driver 11 - heart rate vs sound level

Figure 4.13 shows the truck driver 11 started his journey from location 23.922381, 90.435189 to 23.906602, 90.414957 and time from 6/24/2019 8:23:47 am to 6/24/2019 8:37:18 am. Here, we measured the time in seconds (s), heart rate in beats per minutes (bpm) and sound level in decibel (dB). Here, sound range was 86 to 90 db and heart rate range was 88 to 99 bpm. We examined, when sound level is increased then heart rate also increased.

Figure 4.14 shows the bus driver 12 started his journey from location 23.874251, 90.400337 to 23.874764, 90.400676 and time from 6/25/2019 6:58:54 pm to 6/25/2019 7:07:30 pm. Here, we measured the time in seconds (s), heart rate in beats per minutes (bpm) and sound level in decibel (dB). Here, sound range was 85 to 88 db and heart rate range was 87 to 92 bpm. We

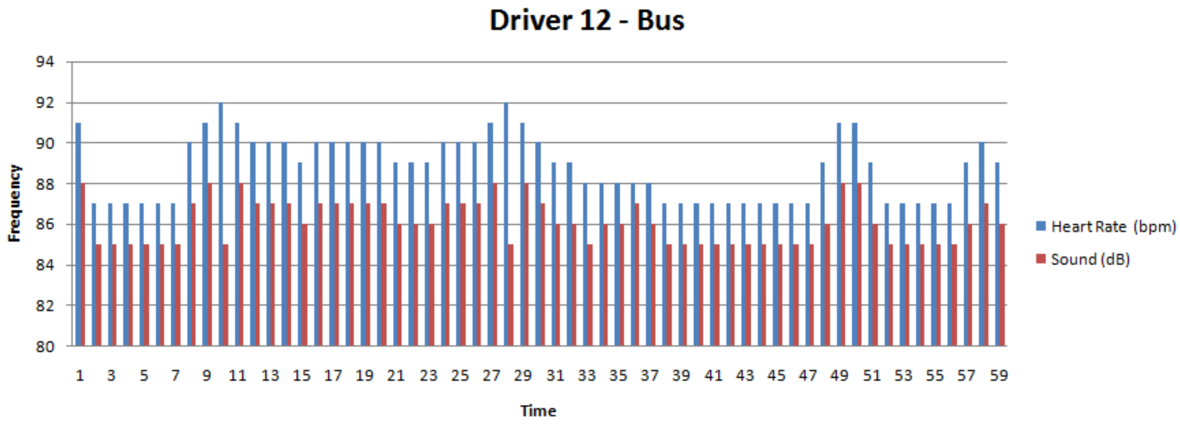


Figure 4.14: Bus driver 12 - heart rate vs sound level

examined, when sound level is increased then heart rate also increased.

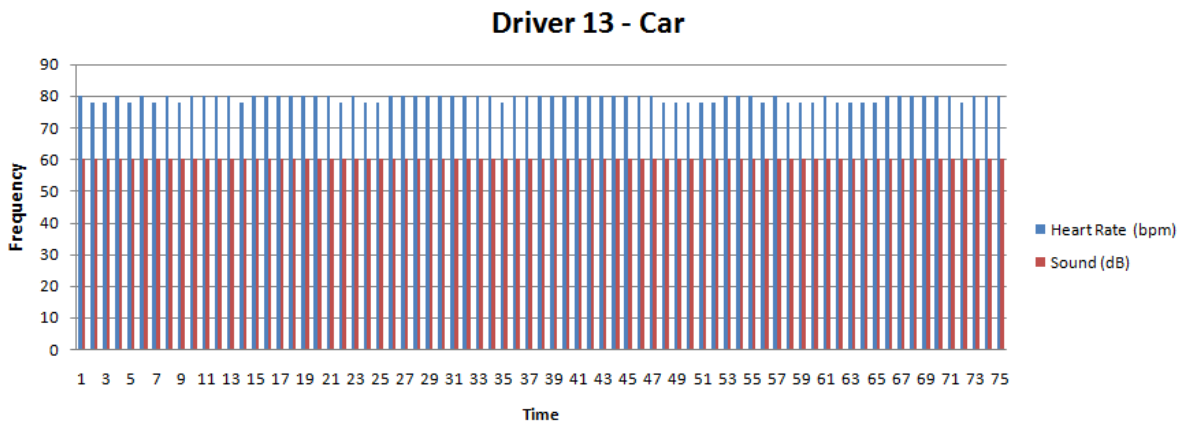


Figure 4.15: AC Car driver 13 - heart rate vs sound level

Figure 4.15 shows the ac car driver 13 started his journey from location 23.891536, 90.400591 to 23.94245, 90.436732 and time from 6/23/2019 7:55:35 pm to 6/23/2019 8:23:10 pm. Here, we measured the time in seconds (s), heart rate in beats per minutes (bpm) and sound level in decibel (dB). Here, sound range was 60 to 60 db and heart rate range was 78 to 80 bpm. We examined, sound level is normal and heart rate is also normal.

Figure 4.16 shows the bus driver 14 started his journey from location 23.863876, 90.401109 to 23.864879, 90.421106 and time from 6/18/2019 7:20:27 pm to 6/18/2019 7:30:33 pm. Here, we measured the time in seconds (s), heart rate in beats per minutes (bpm) and sound level in decibel (dB). Here, sound range was 81 to 90 db and heart rate range was 90 to 97 bpm. We examined, when sound level is increased then heart rate also increased.

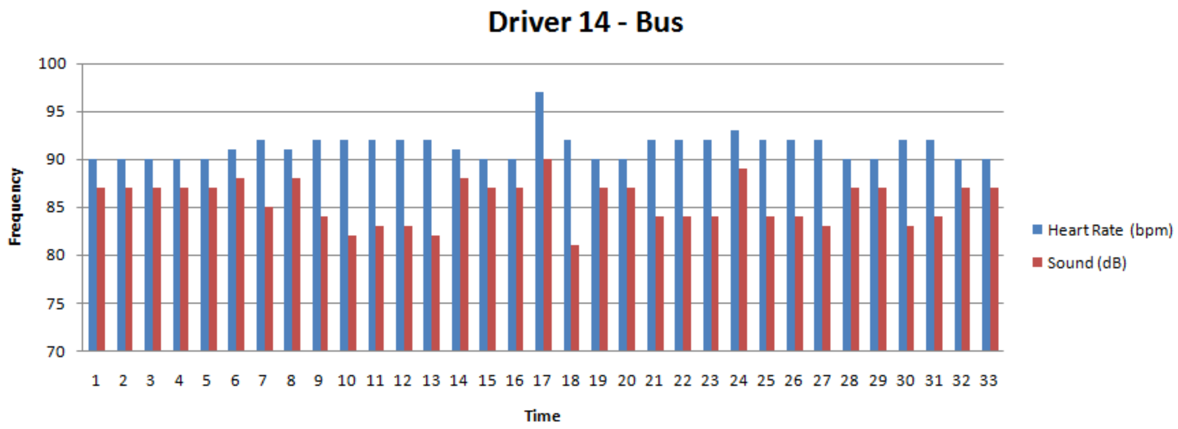


Figure 4.16: Bus driver 14 - heart rate vs sound level

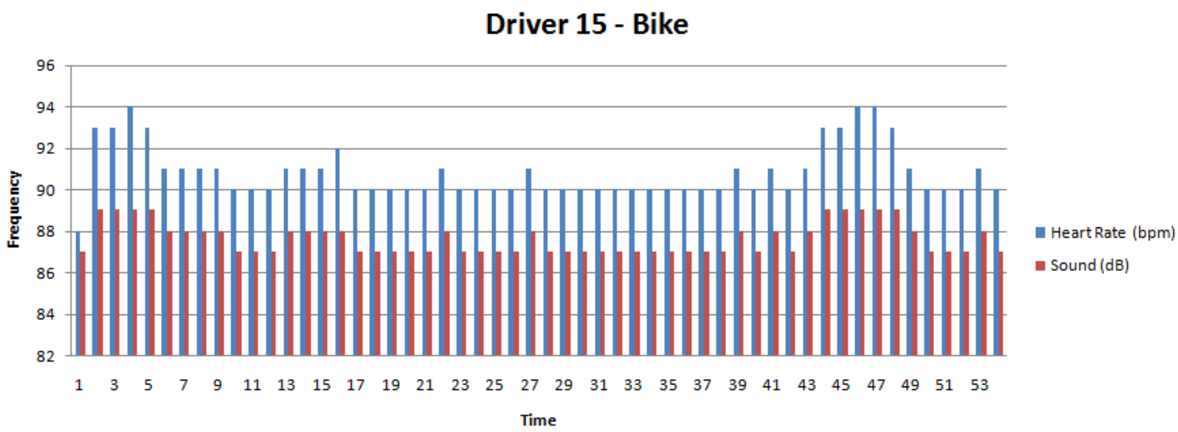


Figure 4.17: Bike driver 15 - heart rate vs sound level

Figure 4.17 shows the bike driver 15 started his journey from location 23.919649, 90.43259 to 23.901083, 90.410153 and time from 6/25/2019 8:46:42 am to 6/25/2019 9:00:54 am. Here, we measured the time in seconds (s), heart rate in beats per minutes (bpm) and sound level in decibel (dB). Here, sound range was 87 to 89 db and heart rate range was 88 to 94 bpm. We examined, when sound level is increased then heart rate is also increased.

Figure 4.18 shows the bus driver 16 started his journey from location 23.906602, 90.414853 to 23.926612, 90.434873 and time from 6/24/2019 8:42:34 am to 6/24/2019 8:57:21 am. Here, we measured the time in seconds (s), heart rate in beats per minutes (bpm) and sound level in decibel (dB). Here, sound range was 86 to 90 db and heart rate range was 88 to 102 bpm. We examined, when sound level is increased then heart rate also increased.

Figure 4.19 shows the bus driver 17 started his journey from location 23.873872, 90.401109 to 23.883877, 90.411189 and time from 6/24/2019 7:20:27 pm to 6/24/2019 7:30:33 pm. Here, we measured the time in seconds (s), heart rate in beats per minutes (bpm) and sound level in

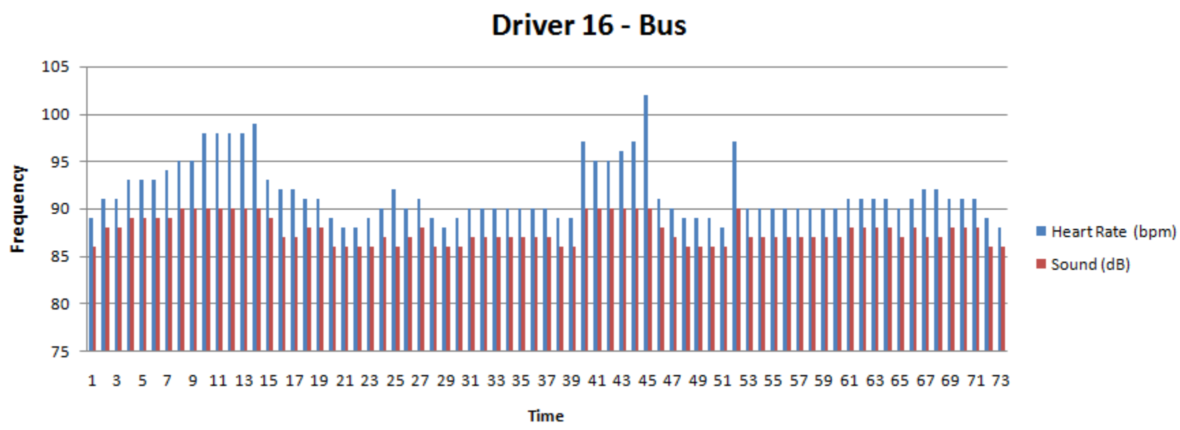


Figure 4.18: Bus driver 16 - heart rate vs sound level

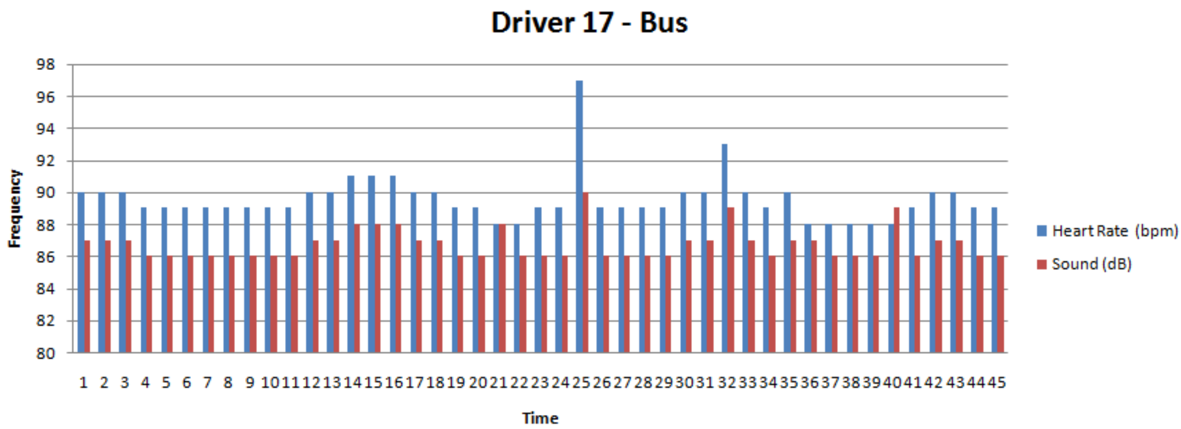


Figure 4.19: Bus driver 17 - heart rate vs sound level

decibel (dB). Here, sound range was 86 to 90 db and heart rate range was 88 to 97 bpm. We examined, when sound level is increased then heart rate also increased.

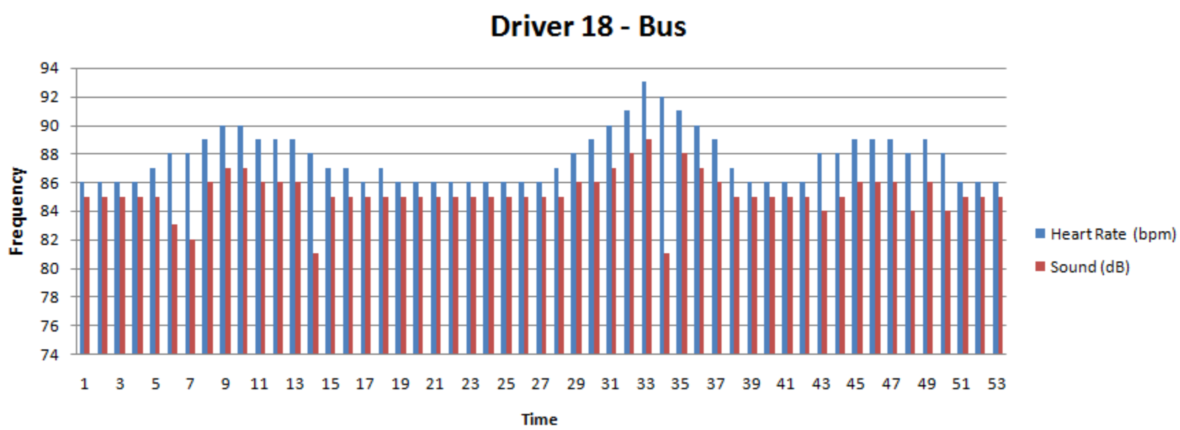


Figure 4.20: Bus driver 18 - heart rate vs sound level

Figure 4.20 shows the bus driver 18 started his journey from location 23.869297, 90.400368 to 23.877546, 90.425593 and time from 6/23/2019 7:28:36 pm to 6/23/2019 7:40:32 pm. Here, we measured the time in seconds (s), heart rate in beats per minutes (bpm) and sound level in decibel (dB). Here, sound range was 81 to 89 db and heart rate range was 86 to 93 bpm. We examined, when sound level is increased then heart rate also increased.

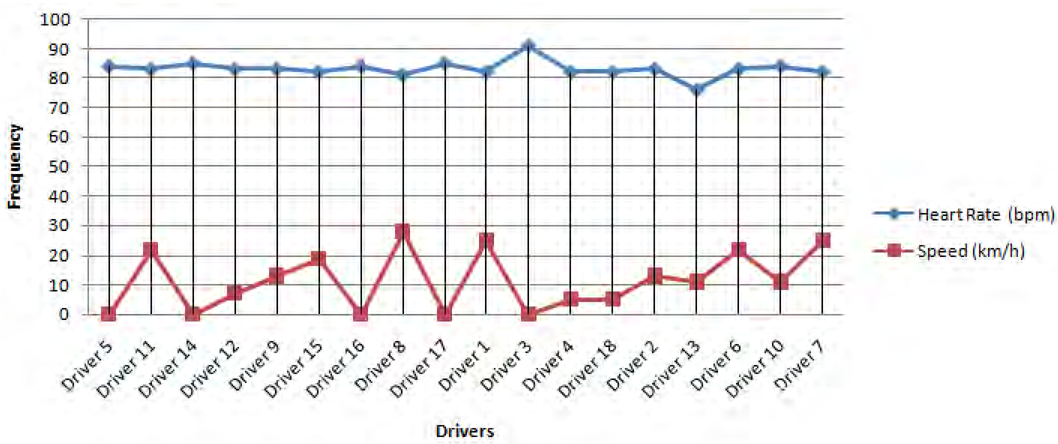


Figure 4.21: Vehicle speed on heart rate of drivers

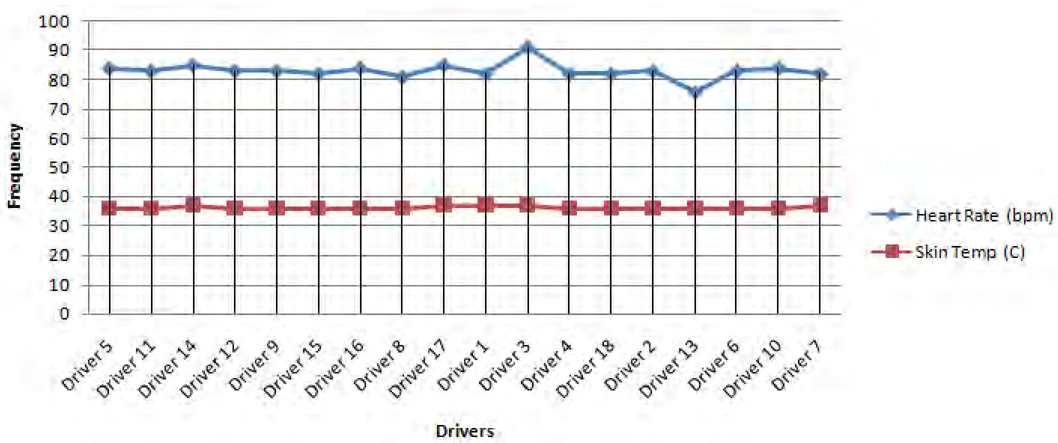


Figure 4.22: Skin temperature on heart rate of drivers

Validation: For validation of the result we have used the standard scale provided by [20]. After the experiment, the result obtained by the system has been compared with the outcome of the drivers feedback. Our system only captured *stressful* and *not stressful* states of drivers.

Drivers Feedback: While designing the system, we asked their opinion about our research, all of our research and field test drivers responded positively regarding necessity of such system. Mainly they appreciated data collection system as it was automated and they liked automatic alert system. They gave feedback. Some of them liked continuous monitoring, alert system and

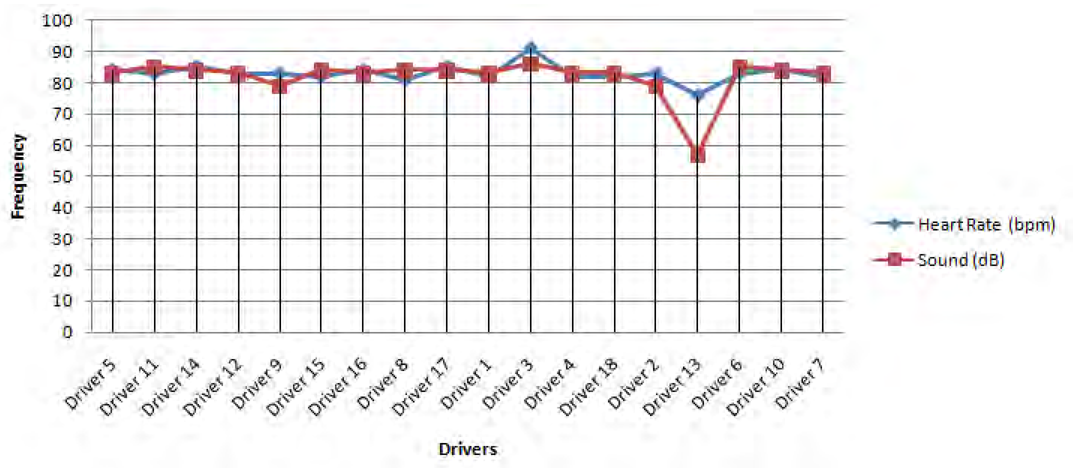


Figure 4.23: Sound on heart rate of drivers

SL	Name	Vehicle Type	Driver Stress Status	System Stress Status	Comparison
1	Driver 1	Bus (AC)	No	Not Stressful	True Negative
2	Driver 2	Leguna	No	Not Stressful	True Negative
3	Driver 3	Bus	No	Not Stressful	True Negative
4	Driver 4	Bike	Yes	Not Stressful	False Positive
5	Driver 5	Bus	Yes	Stressful	True Positive
6	Driver 6	Leguna	No	Not Stressful	True Negative
7	Driver 7	Bus	No	Not Stressful	True Negative
8	Driver 8	Bus	No	Not Stressful	True Negative
9	Driver 9	Car	Yes	Not Stressful	False Positive
10	Driver 10	Truck	Yes	Not Stressful	False Positive
11	Driver 11	Truck	No	Not Stressful	True Negative
12	Driver 12	Bus	Yes	Not Stressful	False Positive
13	Driver 13	Car (AC)	No	Not Stressful	True Negative
14	Driver 14	Bus	No	Not Stressful	True Negative
15	Driver 15	Bike	No	Not Stressful	True Negative
16	Driver 16	Bus	No	Not Stressful	True Negative
17	Driver 17	Bus	No	Not Stressful	True Negative
18	Driver 18	Bus	No	Not Stressful	True Negative

Figure 4.24: Drivers stress state and system generated state

storing data of drivers psychological states. Figure 4.24 is presented as the drivers feedback with psychological states.

Results: The experimental result shows the system has correctly determined the psychological state of the drivers. 22% cases are determined as false positive, 72% cases are determined as true negative and 6% cases are determined as true positive. Several reasons may be thought-about accountable behind this development. The scale used to validate the system requires conscious response regarding one drivers psychological state.

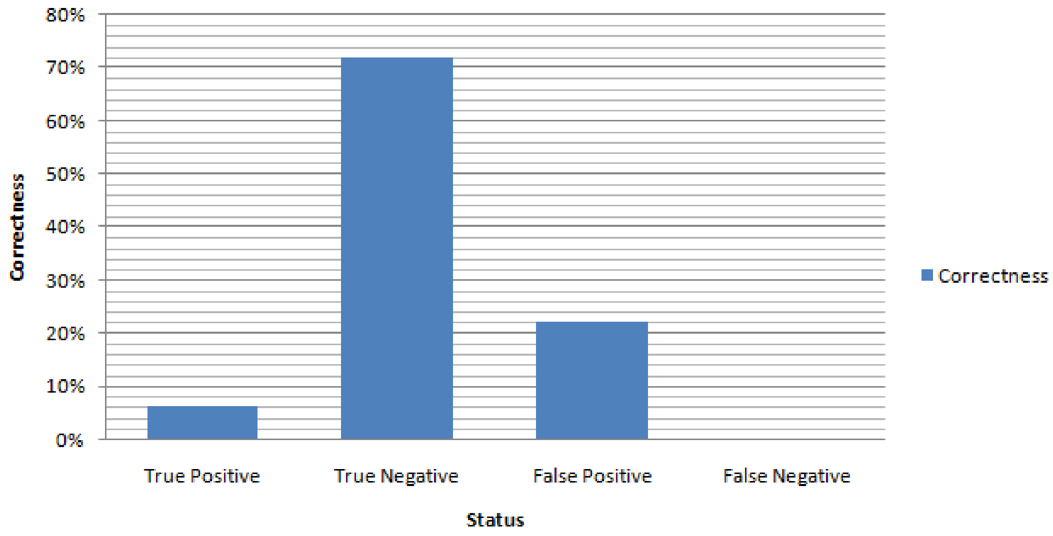


Figure 4.25: Performance evaluation

Stressed individuals are often found to hide their emotional condition. It is also evident that false negative reading is 0% of the experimental cases, which indicates that this system can be reliable to monitor and alert stress individuals. From figures 4.21 - 4.23 show the heart rate, sound level, vehicle speed, and skin temperature condition of the drivers and from figures 4.3 to 4.20 show individual drivers heart rate and sound level condition. Figure 4.25 presents the drivers stress level performance evaluation. We find that in non-AC vehicles, heart rate increases if sound heard is more. In AC vehicles, heart rate varies less and shows the stability.

Chapter 5

Conclusion and Future Direction

The target of this research is to use technology to find the impact of sound pollution on driving. We have tended to detect psychological stress that hampers driving capability. In the beginning, we led a study that uncovers critical relationship between stress and sound level and its harsh impact on drivers. We collected data in different conditions- in sound contaminated condition, in less noisy condition (AC bus or AC car). Using smart wearable system we collected data for identifying drivers physiological condition. We developed an android application for data collection from the wearable smart watches. Stress has been computed gathering RMSSD of Heart Rate Variability of the drivers. The system gives the indications of stress from the sensor data. If there is stress feeling recognized, a programmed alarm is given. In trial arrangement, the framework performed with high accuracy. In future, we will consider different biological and environmental factors for measurement of stress, which may be helpful for better stress management. We will also consider different location route map for identifying the road conditions. Our ultimate goal is to have safe driving, create awareness among drivers to honk less while driving to reduce sound pollution in the road.

References

[1] M. Tasnim, R. Shahriyar, N. Nahar, and H. Mahmud, "Intelligent Depression Detection and Support System: Statistical Analysis, Psychological Review and Design Implication", in Proc. of IEEE 18th International Conference on e-Health Networking, Applications and Services(Healthcom), Munich, Germany, 2016.

[2] N. Munla, M. Khalil, Shahin, and A. Mourad, "Driver Stress Level Detection using HRV Analysis", In Advances in Biomedical Engineering (ICABME), NY, USA, 2015.

[3] B. H. Dalton and D. G. Behm, "Effects of Noise and Music on Human and Task Performance: A Systematic Review", Occupational Argonomics, vol. 7, issue 3, pp. 143-152, 2007.

[4] X. Yu, "Real-time Nonintrusive Detection of Driver Drowsiness", Technical Report of ITS, University of Minnesota, USA, 2009.

[5] J. Vicente, P. Laguna, A. Bartra and R. Bailon, "Detection of Driver's Drowsiness by Means of HRV Analysis", Computing in Cardiology, vol. 38, pp. 89-92, 2011.

[6] J. A. Healey and R. W. Picard, "Detecting Stress During Real-World Driving Tasks Using Physiological Sensors", IEEE Transaction of ITS, vol. 6, issue 2, pp. 156-166, 2005.

[7] R. Roy and K. Venkatasubramanian, "EKG/ECG Based Driver Alert System for Long Haul Drive", Indian Journal of Science and Technology, vol. 8, issue 19, 2015.

[8] M. Miyaji, "Method of Drowsy State Detection for Driver Monitoring Function", International Journal of Information and Electronics Engineering, vol. 4, issue 4, pp. 264-268, 2014.

[9] T. A. Zografos and D. G. Katritsis, "Guidelines and Regulations for Driving in Heart Disease", Hellenic Journal Cardiology, vol. 52, issue 3, pp. 226-234, 2010.

[10] M. Salai, I. Vassanyi, and I. Kosa, "Stress Detection Using Low Cost Heart Rate Sensors", Journal of Health care Engineering, Hindawi, vol. 2016, pp. 1-13, 2016.

[11] Microsoft Band Features. https://www.microsoft.com/microsoft-band/en-us/features.

[12] D. Pramendra, and S. Vartika, "Environmental Noise Pollution Monitoring and Impacts On Human Health in Dehradun City, Uttarakhand, India", Civil and Environmental Research, vol. 1, issue 1, 2011.

[13] M. J. Aslam, M. A. Aslam, and A. Batool, "Effect of noise pollution on hearing of public transport drivers in Lahore city", Pak J Med Sci, vol. 24, issue 1, pp. 142-146, 2008.

[14] HeartRate. Depression and heart disease : Mind and mood affect the heart. http://www.health.harvard.edu/press-releases/depression-and-heart-disease.

[15] U Rajendra Acharya, K Paul Joseph, Natarajan Kannathal, Choo Min Lim, and Jasjit S Suri, "Heart rate variability: a review", Medical and biological engineering and computing, vol. 44, issue 12, pp. 1031–1051, 2006.

[16] Alberts Aldersons and Andris Buikis "Mathematical algorithm for heart rate variability analysis", In Proceedings of the 11th WSEAS international conference on Applied informatics and communications, and Proceedings of the 4th WSEAS International conference on Biomedical electronics and biomedical informatics, and Proceedings of the international conference on Computational engineering in systems applications. World Scientific and Engineering Academy and Society (WSEAS), pp. 381–386, 2011.

[17] Frequency Domain Measures. Frequency domain measures: The fourier transform, the lomb periodogram, and other methods. https://www.physionet.org/events/hrv-2006/moody-1.html.

[18] George B Moody, "Spectral analysis of heart rate without resampling", In Computers in Cardiology 1993, Proceedings, IEEE, pp. 715–718, 1993.

[19] Marcus W Agelink, Cavit Boz, Heiko Ullrich, and Jurgen Andrich, "Relationship between major depression and heart rate variability.: Clinical consequences and implications for antidepressive treatment", vol. 113, issue 1, pp. 139–149, 2002.

[20] MD Uddin and MM Rahman, "Development of a scale of depression for use in bangladesh", Bangladesh psychological Studies, vol. 15, pp. 25–44, 2005.

[21] Kusuma Kumari B.M, "A Real Time Driver Drowsiness Detection System", International Conference on Information and Communication Technologies (ICICT), 2014.

Chapter 6

Codes

6.1 Database Helper Class

```
public class DatabaseHelper extends SQLiteOpenHelper {
    public static final String DB_NAME = "dbDriverStress.db";
    public static final String TB_NAME = "tbDriverStressAll";
    public static final String TB_NAME_ANA = "tbDriverStressAna";
    public static final String col_u_id = "u_id";
    public static final String col_u_name = "u_name";
    public static final String col_u_age = "u_age";
    public static final String col_u_gender = "u_gender";
    public static final String col_u_vehicleType = "u_vehicleType";
    public static final String col_u_heartRate = "u_heartRate";
    public static final String col_u_hrv = "u_hrv";
    public static final String col_u_skinTemp = "u_skinTemp";
    public static final String col_u_decibel = "u_decibel";
    public static final String col_u_latitude = "u_latitude";
    public static final String col_u_longitude = "u_longitude";
    public static final String col_u_distance = "u_distance";
    public static final String col_u_speed = "u_speed";
    public static final String col_u_datetime = "u_datetime";
    public static final String col_u_id1 = "u_id";
    public static final String col_u_name1 = "u_name";
    public static final String col_u_age1 = "u_age";
    public static final String col_u_gender1 = "u_gender";
    public static final String col_u_vehicleType1 = "u_vehicleType";
    public static final String col_u_avgHeartRate1 = "u_avgHeartRate";
    public static final String col_u_rmssd1 = "u_rmssd";
    public static final String col_u_avgSkinTemp1 = "u_avgSkinTemp";
    public static final String col_u_avgSound1 = "u_avgSound";
    public static final String col_u_avgDistance1 = "u_avgDistance";
    public static final String col_u_avgSpeed1 = "u_avgSpeed";
```

```
    public static final String col_u_datetime1 = "u_datetime";
    public static final String col_u_alarm1 = "u_alarm";

    public DatabaseHelper(Context context) {
        super(context, DB_NAME, null, 1);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE IF NOT EXISTS "
            + TB_NAME
            + "("
            + "u_id INTEGER PRIMARY KEY AUTOINCREMENT,"
            + "u_name TEXT,"
            + "u_age TEXT,"
            + "u_gender TEXT,"
            + "u_vehicleType TEXT,"
            + "u_heartRate TEXT,"
            + "u_hrv TEXT,"
            + "u_skinTemp TEXT,"
            + "u_decibel TEXT,"
            + "u_latitude TEXT,"
            + "u_longitude TEXT,"
            + "u_distance TEXT,"
            + "u_speed TEXT,"
            + "u_datetime TEXT"
            + ")");

        db.execSQL("CREATE TABLE IF NOT EXISTS "
            + TB_NAME_ANA
            + "("
            + "u_id INTEGER PRIMARY KEY AUTOINCREMENT,"
            + "u_name TEXT,"
            + "u_age TEXT,"
            + "u_gender TEXT,"
            + "u_vehicleType TEXT,"
            + "u_avgHeartRate TEXT,"
            + "u_rmssd TEXT,"
            + "u_avgSkinTemp TEXT,"
            + "u_avgSound TEXT,"
            + "u_avgDistance TEXT,"
            + "u_avgSpeed TEXT,"
            + "u_datetime TEXT,"
            + "u_alarm TEXT"
            + ")");
    }
```

```

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + TB_NAME);
        db.execSQL("DROP TABLE IF EXISTS " + TB_NAME_ANA);
        onCreate(db);
    }

    public boolean insertData(String u_name, String u_age, String u_gender, String u_vehicleType,
  String u_heartRate, String u_hrv, String u_skinTemp, String u_decibel, String u_latitude,
  String u_longitude, String u_distance, String u_speed, String u_datetime) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(col_u_name, u_name);
        contentValues.put(col_u_age, u_age);
        contentValues.put(col_u_gender, u_gender);
        contentValues.put(col_u_vehicleType, u_vehicleType);
        contentValues.put(col_u_heartRate, u_heartRate);
        contentValues.put(col_u_hrv, u_hrv);
        contentValues.put(col_u_skinTemp, u_skinTemp);
        contentValues.put(col_u_decibel, u_decibel);
        contentValues.put(col_u_latitude, u_latitude);
        contentValues.put(col_u_longitude, u_longitude);
        contentValues.put(col_u_distance, u_distance);
        contentValues.put(col_u_speed, u_speed);
        contentValues.put(col_u_datetime, u_datetime);
        long result = db.insert(TB_NAME, null, contentValues);
        if(result == -1)
        {
            return false;
        }
        else
        {
            return true;
        }
    }

    public void insertDataAna(String u_name, String u_age, String u_gender, String u_vehicleType,
  String u_avgHeartRate, String u_rmssd, String u_avgSkinTemp, String u_avgSound,
  String u_avgDistance, String u_avgSpeed, String u_datetime, String
```

```
    u_alarm) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(col_u_name1, u_name);
        contentValues.put(col_u_age1, u_age);
        contentValues.put(col_u_gender1, u_gender);
        contentValues.put(col_u_vehicleType1, u_vehicleType);
        contentValues.put(col_u_avgHeartRate1, u_avgHeartRate);
        contentValues.put(col_u_rmssd1, u_rmssd);
        contentValues.put(col_u_avgSkinTemp1, u_avgSkinTemp);
        contentValues.put(col_u_avgSound1, u_avgSound);
        contentValues.put(col_u_avgDistance1, u_avgDistance);
        contentValues.put(col_u_avgSpeed1, u_avgSpeed);
        contentValues.put(col_u_datetime1, u_datetime);
        contentValues.put(col_u_alarm1, u_alarm);
        db.insert(TB_NAME_ANA, null, contentValues);
    }

    public Cursor getAllData() {
        SQLiteDatabase db = this.getWritableDatabase();
        Cursor res = db.rawQuery("select * from " + TB_NAME + " order by "
    +
    col_u_id + " desc", null);
        return res;
    }

    public Cursor getAllDataAna() {
        SQLiteDatabase db = this.getWritableDatabase();
        Cursor res = db.rawQuery("select * from " + TB_NAME_ANA + " order
    by " +
    col_u_id + " desc", null);
        return res;
    }

    public void deleteDataAll () {
        SQLiteDatabase db = this.getWritableDatabase();
        db.execSQL("delete from " + TB_NAME);
    }

    public void deleteDataAna () {
        SQLiteDatabase db = this.getWritableDatabase();
        db.execSQL("delete from " + TB_NAME_ANA);
    }

    public Cursor getAllDataByLimit(int limit) {
        SQLiteDatabase db = this.getWritableDatabase();
```

```
        Cursor res = db.rawQuery("select * from " + TB_NAME + " order by "
    + col_u_id +
    " desc limit " + limit, null);
        return res;
    }
}
```

6.2 Location Helper Class

```
public class GetLocation extends Service implements LocationListener {
    private final Context mContext;
    boolean isGPSEnabled = false;
    boolean isNetworkEnabled = false;
    boolean canGetLocation = false;
    private static final long MIN_DISTANCE_CHANGE_FOR_UPDATES = 1; //1
    meter
    private static final long MIN_TIME_BW_UPDATES = 3000; //3 seconds
    protected LocationManager locationManager;
    Location location;
    double latitude;
    double longitude;

    public GetLocation(Context context) {
        this.mContext = context;
        getLocation();
    }

    public Location getLocation() {
        try {
            locationManager = (LocationManager) mContext.getSystemService(
    LOCATION_SERVICE);
            isGPSEnabled = locationManager.isProviderEnabled(
    LocationManager.GPS_PROVIDER);
            isNetworkEnabled = locationManager.isProviderEnabled(
    LocationManager.NETWORK_PROVIDER);

            if (!isGPSEnabled && !isNetworkEnabled) {
              this.canGetLocation = false;
            }
            else {
                this.canGetLocation = true;
                if (isNetworkEnabled) {
                    locationManager.requestLocationUpdates(LocationManager.
          NETWORK_PROVIDER, MIN_TIME_BW_UPDATES,
    MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
                    if (locationManager != null) {
```

```
                    location = locationManager.getLastKnownLocation(
LocationManager.NETWORK_PROVIDER);
                    if (location != null) {
                        latitude = location.getLatitude();
                        longitude = location.getLongitude();
                    }
                }
            }

            if (isGPSEnabled) {
                if (location == null) {
                  locationManager.requestLocationUpdates(
LocationManager.
         GPS_PROVIDER, MIN_TIME_BW_UPDATES,
MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
                    if (locationManager != null) {
                        location = locationManager.getLastKnownLocation
(LocationManager.GPS_PROVIDER);
                        if (location != null) {
                            latitude = location.getLatitude();
                            longitude = location.getLongitude();
                        }
                    }
                }
            }
        }
    }
    catch (Exception e) {
    }
    return location;
}

public void stopUsingGPS(){
    if(locationManager != null){
        locationManager.removeUpdates(GetLocation.this);
    }
}

public double getLatitude(){
    if(location != null){
        latitude = location.getLatitude();
    }
    return latitude;
}

public double getLongitude(){
```

```
        if(location != null){
            longitude = location.getLongitude();
        }
        return longitude;
    }

    public boolean canGetLocation() {
        return this.canGetLocation;
    }

    public void showSettingsAlert(){
        AlertDialog.Builder alertDialog = new AlertDialog.Builder(mContext)
;
        alertDialog.setTitle("GPS is settings");
        alertDialog.setMessage("GPS is not enabled. Do you want to go to
settings menu?");

        alertDialog.setPositiveButton("Settings", new DialogInterface.
OnClickListener() {
            public void onClick(DialogInterface dialog,int which) {
                Intent intent = new Intent(Settings.
ACTION_LOCATION_SOURCE_SETTINGS);
                mContext.startActivity(intent);
            }
        });

        alertDialog.setNegativeButton("Cancel", new DialogInterface.
OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
              dialog.cancel();
            }
        });

        alertDialog.show();
    }

    @Override
    public IBinder onBind(Intent arg0) {
        return null;
    }
}
```

6.3 Profile Class

```
public class StartActivity extends Activity implements
    OnItemSelectedListener {
```

```
EditText etName, etAge;
Spinner spVehicleType;
  Button btNext, btViewData, btExportData, btDeleteData;
String Name, Age, VehicleType;
RadioGroup genderRadioGroup;
DatabaseHelper myDb;
String[] vehicleTypes;
ArrayAdapter<String> adapter;

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

      setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
      etName = (EditText) findViewById(R.id.etName);
      etAge = (EditText) findViewById(R.id.etAge);
      spVehicleType = (Spinner) findViewById(R.id.spVehicleType);
      btNext = (Button) findViewById(R.id.btNext);
      btViewData = (Button) findViewById(R.id.btViewData);
      btExportData = (Button) findViewById(R.id.btExportData);
      btDeleteData = (Button) findViewById(R.id.btDeleteData);
      genderRadioGroup = (RadioGroup) findViewById(R.id.rgGender);
      myDb = new DatabaseHelper(this);

      vehicleTypes = new String[]{"Car", "Bike", "Bus", "Truck", "Leguna"
 , "CNG"};
      adapter = new ArrayAdapter<String>(this, android.R.layout.
  simple_spinner_dropdown_item, vehicleTypes);
      spVehicleType.setAdapter(adapter);
      spVehicleType.setOnItemSelectedListener(this);

      btNext.setOnClickListener(new View.OnClickListener() {
          public void onClick(View v) {
            Name = etName.getText().toString();
          Age = etAge.getText().toString();
          if(!Name.trim().isEmpty() && !Age.trim().isEmpty()){
            Intent intent = new Intent(getApplicationContext(),
 MainActivity.class);
              intent.putExtra("Name", Name);
              intent.putExtra("Age", Age);
              int id = genderRadioGroup.getCheckedRadioButtonId();
                  RadioButton radioButton = (RadioButton) findViewById(id
 );
                  intent.putExtra("Gender", radioButton.getText().
 toString());
```

```
                    intent.putExtra("VehicleType", VehicleType);
                startActivity(intent);
            }
            else {
              Toast.makeText(getBaseContext(), "Data required", Toast.
LENGTH_SHORT).show();
            }
            }
        });

        btViewData.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
              Cursor res = myDb.getAllData();
                if(res.getCount() == 0) {
                  Toast.makeText(getBaseContext(), "Nothing found", Toast.
LENGTH_SHORT).show();
                    return;
                }
                StringBuffer buffer = new StringBuffer();
                while (res.moveToNext()) {
                  buffer.append("Name: "+ res.getString(1)+"\n");
                  buffer.append("Age: "+ res.getString(2)+"\n");
                  buffer.append("Gender: "+ res.getString(3)+"\n");
                  buffer.append("Vehicle Type: "+ res.getString(4)+"\n");
                  buffer.append("Heart Rate: "+ res.getString(5)+"\n");
                  buffer.append("HRV: "+ res.getString(6)+"\n");
                  buffer.append("Skin Temp: "+ res.getString(7)+"\n");
                  buffer.append("Decibel: "+ res.getString(8)+"\n");
                  buffer.append("Latitude: "+ res.getString(9)+"\n");
                  buffer.append("Longitude: "+ res.getString(10)+"\n");
                  buffer.append("Distance: "+ res.getString(11)+"\n");
                  buffer.append("Speed: "+ res.getString(12)+"\n");
                  buffer.append("Datetime: "+ res.getString(13)+"\n\n");
                }
                showMessage("Data",buffer.toString());
            }
      });

        btExportData.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
              ExportData();
              ExportAnalysis();
            }
    });
```

```java

        btDeleteData.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
              myDb.deleteDataAll();
              myDb.deleteDataAna();
              Toast.makeText(getBaseContext(), "Delete Successfull", Toast.
  LENGTH_SHORT).show();
            }
    });
  }

protected void ExportData(){
    File exportDir = new File(Environment.getExternalStorageDirectory(), ""
  );
        if (!exportDir.exists()) {
            exportDir.mkdirs();
        }
        File file = new File(exportDir, "export-data.csv");
        try {
            file.createNewFile();
            PrintWriter csvWrite = new PrintWriter(new FileWriter(file));
            Cursor curCSV = myDb.getAllData();
            csvWrite.println("Name"+","+"Age"+","+"Gender"+","+"Vehicle
  Type"+","+
      "Heart Rate"+","+"HRV"+","+"Skin Temp"+","+"Decibel"+","+"Latitude"+"
  ,"+
      "Longitude"+","+"Distance"+","+"Speed"+","+"Time");
            while (curCSV.moveToNext()) {
                csvWrite.println(curCSV.getString(1)+","+curCSV.getString
  (2)+","+
        curCSV.getString(3)+","+curCSV.getString(4)+","+curCSV.getString(5)
  +","+
        curCSV.getString(6)+","+curCSV.getString(7)+","+curCSV.getString(8)
  +","+
        curCSV.getString(9)+","+curCSV.getString(10)+","+curCSV.getString
  (11)+","+
        curCSV.getString(12)+","+curCSV.getString(13));
            }
            csvWrite.close();
            curCSV.close();
            Toast.makeText(getBaseContext(), "Export All Data Successfull",
   Toast.LENGTH_SHORT).show();
        }
        catch(Exception ex) {
          Toast.makeText(getBaseContext(), ex.getMessage(), Toast.
```

```
 LENGTH_SHORT).show();
        }
  }

  protected void ExportAnalysis(){
    File exportDir = new File(Environment.getExternalStorageDirectory(), ""
 );
        if (!exportDir.exists()) {
            exportDir.mkdirs();
        }
        File file = new File(exportDir, "analysis-data.csv");
        try {
            file.createNewFile();
            PrintWriter csvWrite = new PrintWriter(new FileWriter(file));
            Cursor curCSV = myDb.getAllDataAna();
            csvWrite.println("Name"+","+"Age"+","+"Gender"+","+"Vehicle
 Type"+","+
    "AVG Heart Rate"+","+"LOGRMSSD"+","+"AVG Skin Temp"+","+"AVG Decibel"
 +","+
    "AVG Distance"+","+"AVG Speed"+","+"Time"+","+"Alarm");
            while (curCSV.moveToNext()) {
              csvWrite.println(curCSV.getString(1)+","+curCSV.getString(2)+
 ","+
        curCSV.getString(3)+","+curCSV.getString(4)+","+curCSV.getString(5)
 +","+
        curCSV.getString(6)+","+curCSV.getString(7)+","+curCSV.getString(8)
 +","+
        curCSV.getString(9)+","+curCSV.getString(10)+","+curCSV.getString
 (11)+","+
        curCSV.getString(12));
            }
            csvWrite.close();
            curCSV.close();
            Toast.makeText(getBaseContext(), "Export Analysis Data
 Successfull",
    Toast.LENGTH_SHORT).show();
        }
        catch(Exception ex) {
          Toast.makeText(getBaseContext(), ex.getMessage(), Toast.
 LENGTH_SHORT).show();
        }
  }

  public void showMessage(String title, String Message){
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setCancelable(true);
```

```
            builder.setTitle(title);
            builder.setMessage(Message);
            builder.show();
        }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.start, menu);
        return true;
    }

    public void onItemSelected(AdapterView<?> adapterView, View v, int position, long id) {
        VehicleType = String.valueOf(adapterView.getItemAtPosition(position));
    }

    public void onNothingSelected(AdapterView<?> parent) {

    }

    @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            switch(item.getItemId()){
            case R.id.action_exit:
              moveTaskToBack(true);
              android.os.Process.killProcess(android.os.Process.myPid());
                System.exit(1);
              return true;
            }
        return(super.onOptionsItemSelected(item));
        }
}
```

6.4 Main Program Class

```
public class MainActivity extends Activity {

    private TextView HRVal;
    private TextView TempVal;
    private TextView BandDecibel;
    private TextView BandLatitude;
    private TextView BandLongitude;
    private Button btStartConsent, btStartHeartRate;
    private BandClient client;
    private String rrInterval;

```

```
private BandHeartRateEventListener mHeartRateEventListener = new
 BandHeartRateEventListener() {
     @Override
     public void onBandHeartRateChanged(final BandHeartRateEvent event)
 {
         if (event != null) {
           HRVal.post(new Runnable() {
             @Override
             public void run() {
               HRVal.setText(String.format("%d", event.getHeartRate()));
             }
         });
         }
     }
 };

 private BandSkinTemperatureEventListener skinTemperatureEventListener =
  new BandSkinTemperatureEventListener() {
     @Override
     public void onBandSkinTemperatureChanged(final
 BandSkinTemperatureEvent event) {
       if (event != null) {
         TempVal.post(new Runnable() {
             @Override
             public void run() {
               TempVal.setText(String.format("%.2f",event.getTemperature
 ()));
             }
         });
         }
     }
 };

 private BandRRIntervalEventListener rrIntervalEventListener = new
 BandRRIntervalEventListener() {
     @Override
     public void onBandRRIntervalChanged(final BandRRIntervalEvent event
 ) {
         if (event != null) {
           rrInterval = String.format("%.6f", event.getInterval());
         }
     }
 };

 private DatabaseHelper myDb;
 private String name, age, gender, vehicleType, date;
```

```
    private double latitude = 0, longitude = 0, latitude_old = 0,
   longitude_old = 0;
    private Calendar calander;
    private SimpleDateFormat simpledateformat;
    private GetLocation gps;
    private MediaRecorder mRecorder;
    private Thread thread_save;
    private Thread thread_update;
    private MediaPlayer mPlayer;
    private int status = 0;

  protected void UpdateAlarm() {
    try {
            if (mPlayer.isPlaying()) {
              mPlayer.stop();
              mPlayer.release();
              mPlayer = MediaPlayer.create(getBaseContext(), R.raw.danger);
            }
            mPlayer.start();
        }
    catch(Exception e) {
      appendToUI(e.getMessage());
        }
    }

  protected void UpdateGPS() {
    gps = new GetLocation(getBaseContext());
        if(gps.canGetLocation()){
          latitude = gps.getLatitude();
          longitude = gps.getLongitude();
            BandLatitude.setText(String.valueOf(latitude));
            BandLongitude.setText(String.valueOf(longitude));
        }
        else{
            gps.showSettingsAlert();
        }
    }

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

        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON
   );
```

```
        HRVal = (TextView) findViewById(R.id.tvHRValue);
      TempVal = (TextView) findViewById(R.id.tvSkinTemp);
        BandDecibel = (TextView) findViewById(R.id.tvDecibel);
        BandLatitude = (TextView) findViewById(R.id.tvLatitude);
        BandLongitude = (TextView) findViewById(R.id.tvLongitude);
        btStartConsent = (Button) findViewById(R.id.btStartConsent);
        btStartHeartRate = (Button) findViewById(R.id.btStartHeartRate);
        mPlayer = MediaPlayer.create(getBaseContext(), R.raw.danger);

        btStartHeartRate.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        new HeartRateSubscriptionTask().execute();
      }
    });

        final WeakReference<Activity> reference = new WeakReference<
   Activity>(this);

        btStartConsent.setOnClickListener(new View.OnClickListener() {
          @SuppressWarnings("unchecked")
            @Override
      public void onClick(View v) {
        new HeartRateConsentTask().execute(reference);
      }
    });

        Intent intent = getIntent();
        name = intent.getStringExtra("Name");
        age = intent.getStringExtra("Age");
        gender = intent.getStringExtra("Gender");
        vehicleType = intent.getStringExtra("VehicleType");
        myDb = new DatabaseHelper(this);

        UpdateGPS();
        UpdateDecibel();

        thread_save = new Thread() {
            @Override
            public void run() {
                try {
                    while (!isInterrupted()) {
                        Thread.sleep(2000);
                        runOnUiThread(new Runnable() {
                            @SuppressLint("SimpleDateFormat")
                            @Override
```

```
                        public void run() {

                            calander = Calendar.getInstance();
                            simpledateformat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
                            date = simpledateformat.format(calander.getTime());
                            UpdateGPS();
                            UpdateDecibel();

                            latitude_old = GetDatabaseValue(9);
                            longitude_old = GetDatabaseValue(10);
                            double distance_m = CalculateDistance(latitude, longitude, latitude_old, longitude_old);
                            double speed_ms = distance_m / 3;
                        double speed_kmh = speed_ms * 3.6;

                            if(!String.valueOf(HRVal.getText()).equals("0")){

                              boolean isInserted = myDb.insertData(name, age, gender, vehicleType, String.valueOf(HRVal.getText()), rrInterval, String.valueOf(TempVal.getText()), String.valueOf(BandDecibel.getText()), String.format("%.6f", latitude), String.format("%.6f", longitude), String.format("%.4f", distance_m), String.format("%.4f", speed_kmh), date);

                              if(isInserted == true){
                                  status = 1;
                                }
                              else {
                                status = 0;
                              }
                            }
                        }
                    });
                }
            }
            catch (InterruptedException e) {
              appendToUI(e.getMessage());
            }
        }
    };

    thread_save.start();

```

```
thread_update = new Thread() {
    @Override
    public void run() {
        try {
            while (!isInterrupted()) {
                Thread.sleep(40000);
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                      if(status == 1) {
                        int dataLimit = 20;
                        String alarm = "";
                    Cursor res = myDb.getAllDataByLimit(dataLimit);
                    double[] myList = new double[res.getCount()];
                    if(res.moveToFirst()) {
                      for(int i = 0; i < res.getCount(); i++) {
                        myList[i] = Double.parseDouble(res.getString(6));
                        res.moveToNext();
                      }
                    }
                    res.close();
                    double total = 0, rmssd = 0, rmssdFinal = 0;
                    int len = myList.length - 1;
                        for (int j = 0; j <= len; j++) {
                           if(j == len) break;
                           double num1 = myList[j];
                           double num2 = myList[j + 1];
                           double sub = num1 - num2;
                           total += Math.pow(sub, 2);
                        }
                        rmssd = Math.sqrt(total/len);
                        rmssdFinal = rmssd * 100;

                        int avgHearRate = (int)GetAverageValue(5, dataLimit);
                        int avgSound = (int)GetAverageValue(8, dataLimit);
                        int avgSkinTemp = (int)GetAverageValue(7, dataLimit);
                        double avgDistance = GetAverageValue(11, dataLimit);
                        double avgSpeed = GetAverageValue(12, dataLimit);

                        if(rmssdFinal < 1 && avgSound > 85) {
```

```
                    alarm = "alert-stress-sound";
                    UpdateAlarm();
                  }
                  else if(rmssdFinal < 1 && avgSound > 85 && avgSpeed > 60) {
                    alarm = "alert-stress-sound-speed";
                    UpdateAlarm();
                  }
                  else if (rmssdFinal < 1) {
                    alarm = "alert-stress";
                    UpdateAlarm();
                  }
                  else {
                    alarm = "alert-off";
                  }

                  myDb.insertDataAna(name, age, gender, vehicleType, String.valueOf(avgHearRate), String.format("%.6f", rmssdFinal), String.valueOf(avgSkinTemp), String.valueOf(avgSound), String.format("%.4f", avgDistance), String.format("%.4f", avgSpeed), date, alarm);
                              }
                          }
                      });
                    }
                }
                catch (InterruptedException e) {
                  appendToUI(e.getMessage());
                }
            }
        };

        thread_update.start();
    }

    @Override
  protected void onResume() {
    super.onResume();
    StartRecorder();
  }

    @Override
  protected void onPause() {
    super.onPause();
    if (client != null) {

      try {
```

```
        client.getSensorManager().registerHeartRateEventListener(
    mHeartRateEventListener);
      }
      catch (BandIOException e) {
        appendToUI(e.getMessage());
      }
      catch (BandException e) {
        appendToUI(e.getMessage());
      }

      try {
        client.getSensorManager().registerSkinTemperatureEventListener(
    skinTemperatureEventListener);
            }
      catch(BandException e) {
              appendToUI(e.getMessage());
            }

      try {
        client.getSensorManager().registerRRIntervalEventListener(
    rrIntervalEventListener);
            }
      catch (InvalidBandVersionException e) {
        appendToUI(e.getMessage());
      }
      catch(BandException e) {
              appendToUI(e.getMessage());
            }
    }

    StopRecorder();
  }

    @Override
    protected void onDestroy() {
        if (client != null) {
            try {
                client.disconnect().await();
            }
            catch (InterruptedException e) {
              appendToUI(e.getMessage());
            }
            catch (BandException e) {
              appendToUI(e.getMessage());
            }
        }
```

```
            StopAlarm();
            thread_save.interrupt();
            thread_update.interrupt();
            super.onDestroy();
        }

    private class HeartRateSubscriptionTask extends AsyncTask<Void, Void, Void> {
        @Override
        protected Void doInBackground(Void... params) {
            try {
                if (getConnectedBandClient()) {
                    if (client.getSensorManager().getCurrentHeartRateConsent() == UserConsent.GRANTED) {
                        client.getSensorManager().registerHeartRateEventListener(mHeartRateEventListener);
                        client.getSensorManager().registerSkinTemperatureEventListener(skinTemperatureEventListener);
                        client.getSensorManager().registerRRIntervalEventListener(rrIntervalEventListener);
                    }
                    else {
                        appendToUI("You have not given this application consent to access heart rate data yet. Please press the Heart Rate Consent button.\n");
                    }
                }
                else {
                    appendToUI("Band isn't connected. Please make sure bluetooth is on and the band is in range.\n");
                }
            }
            catch (Exception e) {
                appendToUI(e.getMessage());
            }
            return null;
        }
    }

    private class HeartRateConsentTask extends AsyncTask<WeakReference<Activity>, Void, Void> {
        @Override
        protected Void doInBackground(WeakReference<Activity>... params) {
            try {
                if (getConnectedBandClient()) {
                    if (params[0].get() != null) {
```

```
            client.getSensorManager().requestHeartRateConsent(params[0].get(), new HeartRateConsentListener() {
              @Override
              public void userAccepted(boolean consentGiven) {
              }
              });
          }
        }
        else {
          appendToUI("Band isn't connected. Please make sure bluetooth is on and the band is in range.\n");
        }
      }
      catch (BandException e) {
        String exceptionMessage="";
        switch (e.getErrorType()) {
        case UNSUPPORTED_SDK_VERSION_ERROR:
          exceptionMessage = "Microsoft Health BandService doesn't support your SDK Version. Please update to latest SDK.\n";
          break;
        case SERVICE_ERROR:
          exceptionMessage = "Microsoft Health BandService is not available. Please make sure Microsoft Health is installed and that you have the correct permissions.\n";
          break;
        default:
          exceptionMessage = "Unknown error occured: " + e.getMessage() + "\n";
          break;
        }
        appendToUI(exceptionMessage);
      }
      catch (Exception e) {
        appendToUI(e.getMessage());
      }
      return null;
    }
  }

  private boolean getConnectedBandClient() throws InterruptedException, BandException {
    if (client == null) {
      BandInfo[] devices = BandClientManager.getInstance().getPairedBands();
      if (devices.length == 0) {
        appendToUI("Band isn't paired with your phone.\n");
```

```
        return false;
      }
      else{
        client = BandClientManager.getInstance().create(getBaseContext(), devices[0]);
        return false;
      }
    }
    else if (ConnectionState.CONNECTED == client.getConnectionState()) {
      return true;
    }
    else{
      appendToUI("Band is connecting...\n");
      return ConnectionState.CONNECTED == client.connect().await();
    }
  }

  private void appendToUI(final String string) {
    this.runOnUiThread(new Runnable() {
            @Override
            public void run() {
              Toast.makeText(getBaseContext(), string, Toast.LENGTH_SHORT).show();
            }
        });
  }

    @Override
    public void onBackPressed() {
      thread_save.interrupt();
      thread_update.interrupt();
      StopAlarm();
      moveTaskToBack(true);
      android.os.Process.killProcess(android.os.Process.myPid());
        System.exit(1);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch(item.getItemId()){
```

```
        case R.id.action_exit:
          thread_save.interrupt();
          thread_update.interrupt();
          StopAlarm();
          moveTaskToBack(true);
          android.os.Process.killProcess(android.os.Process.myPid());
            System.exit(1);
          return true;
        }
    return(super.onOptionsItemSelected(item));
    }

  public void StartRecorder(){
      if (mRecorder == null)
      {
          mRecorder = new MediaRecorder();
          mRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
          mRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
          mRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
          mRecorder.setOutputFile("/dev/null");
          try
          {
              mRecorder.prepare();
          }
          catch (IOException e) {
            appendToUI(e.getMessage());
          }
          catch (SecurityException e) {
            appendToUI(e.getMessage());
          }
          try
          {
              mRecorder.start();
          }
          catch (SecurityException e) {
            appendToUI(e.getMessage());
          }
      }
  }

  public void StopRecorder() {
      if (mRecorder != null) {
          mRecorder.stop();
          mRecorder.release();
          mRecorder = null;
      }
```

```
}

public void StopAlarm() {
  if (mPlayer.isPlaying()) {
    mPlayer.stop();
    mPlayer.release();
    mPlayer = null;
      }
}

public double GetAmplitude() {
    if (mRecorder != null)
        return mRecorder.getMaxAmplitude();
    else
        return 5;
}

public double GetDecibelValue(){
  return  20 * Math.log10(GetAmplitude());
}

public void UpdateDecibel(){
  int getValue = (int)GetDecibelValue();
  if(getValue > 0)
    BandDecibel.setText(String.valueOf(getValue));
  else
    BandDecibel.setText("40");
}

public double CalculateDistance(double lat1, double lon1, double lat2,
   double lon2) {
  if ((lat1 == lat2) && (lon1 == lon2)) {
    return 0;
  }
  else {
    double Radius = 6378.00;
    double dLat = Math.toRadians(lat2 - lat1);
    double dLon = Math.toRadians(lon2 - lon1);
    double a = Math.sin(dLat / 2) * Math.sin(dLat / 2) + Math.cos(Math.
   toRadians(lat1)) * Math.cos(Math.toRadians(lat2)) * Math.sin(dLon / 2) *
    Math.sin(dLon / 2);
    double c = 2 * Math.asin(Math.sqrt(a));
    return Radius * c;
  }
}

```

```
  public double GetAverageValue(int valuePosition, int dataLimit){
    Cursor res = myDb.getAllDataByLimit(dataLimit);
    double[] myList = new double[res.getCount()];
    int len = myList.length - 1;
    double total = 0.00;
    if(len > 0){
      if(res.moveToFirst()) {
        for(int i = 0; i < res.getCount(); i++) {
          myList[i] = Double.parseDouble(res.getString(valuePosition));
          res.moveToNext();
        }
      }
      res.close();
          for (int j = 0; j <= len; j++) {
            total += myList[j];
          }
          return total/len;
    }
    else
    {
      res.close();
      return 0.00;
    }
  }

  public double GetDatabaseValue(int valuePosition){
        Cursor res = myDb.getAllDataByLimit(1);
        double value = 0;
    if(res.getCount() > 0){
      if(res.moveToFirst()) {
        value = Double.parseDouble(res.getString(valuePosition));
      }
    }
    res.close();
    return value;
  }
}
```