Technical University of Denmark

Mathias Pinto Bonnesen

Mobile Apnea Screening (MAS)

Algorithm Design, Experiment & Validation

Master's Thesis, January 2017.

DTU Electrical Engineering
Department of Electrical Engineering

Rigshospitalet

Glostrup

Mobile Apnea Screening (MAS) - Algorithm Design, Experiment & Validation.

Report written by:
Mathias Pinto Bonnesen (s113918).

Main suporvisor:
Associate Professor MSK, M.Sc,.EE, Ph.D., Helge Bjarup Dissing Sørensen, Department of Electrical Engineering, Technical University of Denmark

Clinical supervisor:
Chief Physician, Professor, M.D. Poul Jennum, Danish Center for Sleep Medicine, Department of Clinical Neurophysiology, Glostrup University Hospital.

DTU Elektro
Technical University of Denmark
2800 Kgs. Lyngby
Denmark

studieadministration@elektro.dtu.dk

Project Period: 22.08.16 - 25.01.17

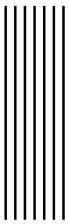
ECTS: 30

Education: Master of Science

Field: Biomedical Engineering

Edition: 1. edition

Remarks: This thesis is submitted as partial fulfillment of the requirements for graduation in the above education at the Technical University of Denmark.



Preface

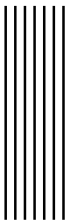
This thesis is submitted as a partial fulfillment of the requirements for a Master of Science degree in Biomedical Engineering at The Technical University of Denmark (DTU).

The project was carried out in coorporation with the Biomedical Signal Processing Group at the Department of Electrical Engineering (DTU) and the Danish Center for Sleep Medicine (DCSM) at Rigshospitalet Glostrup. The work was carried out from August 22th 2016 to January the 25th 2017, with an assigned workload of 30 ECTS points.

The goal of this thesis was to develop a novel method for at-home data acquisition from a subject, during a full night's sleep. To further advance the field of sleep studies, an advanced algorithm for the detection of Obstructive Sleep Apnea (OSA) was developed which used the acquired data. To validate the algorithm, a sleep study was conducted parallel to the proposed method, and the result of categorizing subjects were compared. All data acquisition was carried out at the DCSM, after the experimental design and setup was approved by the regulatory authorities; the Danish Council for Ethics, the Danish Ministry of Health and the Danish Data Protection Agency. The filing for regulatory approval was done by the previous master student Martin Guul, and the Android application used for data acquisition is a modified version of the application he developed in his thesis.

Finally, the project resulted in a scientific publication submitted for review to the IEEE Engineering in Medicine and Biology Society conference 2017. The draft can be found in Appendix A and will be finalized after the submission of the thesis.

Mathias Pinto Bonnesen
Technical University of Denmark (DTU)
January 2017



Acknowledgements

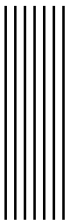
Working on a Master's thesis is a long and demanding process, and I would therefore like to express my gratitude towards a group of people, for their continuing support and kindness towards me and for helping me keeping a high spirit.

First of all, I would like to thank my main supervisor Helge B.D Sørensen, not only for providing me with the opportunity of working on this project, but also for providing valuable and insightfull feedback on our weekly meetings. Next, I would like to thank my clinical supervisor dr. med. Poul Jennum for giving me the opportunity of performing a clinical trial, as well as his knowledgeable inputs and help with setting up the experimental design and discussions of overall clinical aspects of the project.

Furthermore I would like to thank the employees at the Danish Center for Sleep Medicine, Rigshospitalet Glostrup; The Neurofysiology assistants for their involvement in getting subjects for the recordings and the engineers, especially Ph.D. Lykke Kempfner, for helping me with the technical aspect of handling good experimental practice.

I would also like to thank all of my fellow master students and all other members of *The Biomedical Signal Processing Group* at the department of Electrical Engineering (DTU). I would like to thank Martin Guul for his hard and comprehensive ground work when seeking and gaining permission from the necessary regulatory authorities, as well as the work he did on the Android application prior to this project.

Finally, I would like to express my deepest thanks to my family and friends for always being supportive, kind and always ready for a good laugh when i really needed it.



Abstract

Obstructive Sleep Apnea (OSA) is a common sleep disorder estimated to affect 2-4 % of the middle-aged population. It is defined as multiple events with reduced or seized airflow due to an obstruction in the upper airways, and can result in severe complications such as depression and an increased risk of developing cardiovascular diseases. The gold standard diagnostic protocol is a full Polysonography (PSG), which is both a costly and time consuming procedure, due to the conduction in a sleep clinic with manually scoring of results by medical experts. This calls for a more simple solution, ideally, capable of performing the test *at-home*, with a fully automatic analysis of the acquired data, resulting in fast, reliable and uncostly diagnosis.

This study presents a novel *at-home* screening method for OSA using a smartphone, a microphone and a modified armband, to gather continuous biological signals during a whole night's sleep, and a signal processing algorithm to analyze the acquired data. Features extracted are based on event detection algorithms, body movement and demographic information. Four different classifications problems were tested; three binary problems with different thresholds, and one 4-class problem with the classes *healthy, mild-, moderate-* and *severe* OSA. Both a Bagged Decision Tree (BDT) classifier and a Support Vector Machine (SVM) were tested to determine which of the modalities matched the classification problems best to find the most beneficial solution.

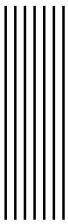
A total of 23 recordings from subjects was used to validate the system, by performing sleep studies parallel to the data acquisition. The sleep study was analyzed by medical experts at the Danish Center for Sleep Medicine (DCSM) Glostrup Rigshospital, and the AHI was used to label each subject into class related to severity of OSA or if they were healthy. The SVM gave the best result in the binary classification problem with *sensitivities* between 85.7 % and 100 %, and accuracies between 78.3 % and 91.3 %. The 4-class classification was not as successful with *sensitivities* between 75 % and 85.7 % and *accuracies* between 56.5 % and 60.9 %. This was thought to be primarily due to difficulties associated with distinguishing between milder forms of OSA, arising from the challenges of correctly labeling *hypapneic* events. However, the method succeeded profoundly as a *proof of concept.*

Resumé

Obstruktiv Søvnapnø (OSA) er en ofte set søvnforstyrrelse som er estimetet til at ramme 2-4 % af den midaldrende befolkning. Den er defineret som *flere events med reduceret eller stop af vejrtræknigner, på grund af obstruktion i de øvre luftveje* hvilket kan resultere i alvorlige komplikationer så som depression og en øget risiko for at udvikle kardiovaskulære sygdomme. Den diagnostiske guldstandard er at foretage en Polysomnografi (PSG), hvilket er både er en dyr og tidskrævende procedure, siden den skal foretages i en søvn klinik og resultaterne skal scores af det medicinske personale. Derfor ville en løsning med mulighed for at optage biologiske signaler *i hjemmet*, og derefter lave en fuldt automatisk undersøgelse af det optagede data for at kunne stille en diagnose, være ideel.

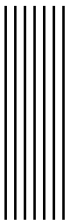
Det her masterprojekt, præsenterer en moderne metode til at screene for OSA *i hjemmet*, ved brug af en smartphone, en mikrofon og et modificeret armbånd, der kan optage biologiske signaler kontinuært over en hel nat samt en avanceret algoritme til at analysere og fortolke data. *Features* udtrukket fra data er baseret på at detektere *events*, undersøge kropslige bevægelser og demografisk data. Fire forskellige klassifikations problemer blev testet; tre binære problemer med forskellige grænseværdier og et 4-klasse problem med klasserne *rask-*, *mild-*, *moderat-* og *alvorlig* OSA. Både Bagged Decision Tree (BDT) såvel som Support Vector Machine (SVM) blev testet for at finde ud af hvilke af de to modaliteter var bedst egnet til brug.

I alt blev 23 optagelser fra forsøgspersoner brugt til at validere systemet, ved at udføre søvnstudier parallelt. Søvnstudierne blev scoret af personalet på Dansk Center for Søvnmedicin (DCSM) på Rigshospitalet Glostrup, og den udregnede Apnea-Hypapnea-Index (AHI) blev brugt til at markere data aht. hvor alvorligt en grad af OSA en forsøgsperson havde, eller om de var raske. Brugen af SVM gav de bedste resultater i den binære klassifikation, med *sensitiviteter* mellem 85.7 % og 91.3 % *accuracy* mellem 78.3 % og 100 %. 4-klasse problemet klarede sig dårligere med *sensitiviteter* mellem 75 % og 85.7 % og *akkuratheder* mellem 56.5 % og 60.9 %. Dette var formegentlig på grund af problemer med korrekt at detektere *hypapneic* events. Alt i alt, lykkedes det at lave en *proof-of-concept* løsning.

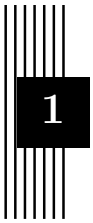


Nomenclature

AHI	Apnea-Hypapnea-Index
AASM	American Association of Sleep Medicine
BMI	Body Mass Index
BDT	Bagged Decision Tree
CRM	Cardio Respiratory Monitoring
CFS	Correlation Feature Selection
CRF	Case Report File
CSA	Central Sleep Apnea
DCSM	Danish Center for Sleep Medicine
ECG	Electroencephalograpy
EEG	Electroencephalography
EMG	Electromyography
EOG	Electrooculography
FFT	Fast Fourie Transform
HR	Heart Rate
HRV	Heart Rate Variability
ICC	Intra-Class Correlation
MEMS	Micro Electro Mechanical Systems
MSA	Mixed Sleep Apnea
ODI	Oxygen Desaturation Index
OSA	Obstructive Sleep Apnea
PCA	Principal Component Analysis
PPG	Photoplethysmography
PSD	Power Spectral Density
PSG	Polysomnography
RR	Respiration Rate
SNR	Signal to Noise Ratio
SRBD	Sleep Related Breathing Disorders
SpO2	Oxygen Saturation
SVM	Support Vector Machine



CONTENTS



1 Introduction

1.1 Motivation

Obstructive Sleep apnea (OSA) is a common sleep disorder estimated to affect 2-4 % of the adult population, with a variation across ethnic groups. The pathological mechanisms responsible for the disorder is a partial or complete seizing of airflow though the upper airway, due to an obstruction [1]. The diagnosis and severity of the disorder is based on the frequency of these events occurring during a night's sleep. Complications related to the disorder ranges from sleepiness, decreased learning ability to more severe complications such as depression and increased risk of developing cardiovascular diseases (ischemic heart disease, stroke etc.).

It is estimated that 90 % of OSA sufferers are undiagnosed [1], which is worrying because the disorder is treatable and early diagnosis can stop the patients from developing the more severe complications. The problem might be the cost and time needed to perform the necessary sleep diagnostic procedures in order to diagnose the disease. This has huge social, health and economical consequences in the society [4], and is therefore an issue that should be addressed.

Furthermore, clinicians at the sleep department of Rigshospitalet Glostrup estimates that approximately 50 % of people referred to their clinic for sleep analysis are in fact healthy, causing a huge loss in medical expanses.

1.2 Problem Statement

The current gold standard is the Polysomnography (PSG), which is a costly and time consuming sleep study, usually conducted at a sleep clinic overnight. Recent developments of Obstructive Sleep Apnea (OSA) diagnostics have been the introduction of the Cardio Respiratory Monitor (CRM), allowing the patients to conduct the analysis at home. Then they only need to drop by the clinic for equipment setup and when the equipment is returned after the sleep study has been conducted.

Even though the CRM is a big step in the right direction, it is still an expensive piece of medical equipment, and the time consuming process of manually scoring each study, is still a challenging problem, when dealing with such a prevalent disorder as OSA. A simple screening method could be of great value, which is why there have been multiple studies focused on creating relative cheap and simple OSA screening solutions [6][21][22].

One approach that has gotten a lot of attention in recent years, is the usage of a smartphone as the mainframe for both the signal acquisition and analysis. The idea is, that if a solution could be created using a smartphone, and a limited amount of other sensors found in most

homes, people would be able to test themselves very easily, potentially leading to the reduction of undiagnosed OSA-sufferers in the general population.

1.3 Project Goals

- Create a simple signal acquisition method, using a smartphone as the mainframe and sensors only found in most homes, capable acquiring biomedical signal measurements for OSA severity assessment. The signal acquisition method should be capable of recording continuously for at least 6 hours, using less than 40 % battery and less than 2 GB data storage.
- Create an advanced screening algorithm capable of using the acquired data to distinguish between healthy subjects and subjects suffering from OSA, as well as further categorize the latter depending on the severity of the disease; mild, moderate and severe OSA.
- Clinically validate the created screening algorithm by performing experimental tests on subjects referred to Rigshospitalet Glostrup for sleep analysis. The classification performance should be validated based on OSA severity, using the performance measures *sensitivity* and *accuracy*. The aim is to reach an accuracy above 80 % and a *sensitivity* above 90 %, based on discussions with the project supervisors.

1.4 Innovation

The field of *at-home* OSA screening using smartphones, is a relatively new field which has gained increased popularity in recent years. This is due to developments in technologies such as more powerful smartphones and cheaper sensors. No clinically validated state of the art studies were found which performed both data acquisition and analysis, using a smartphone as the mainframe. Therefore the proposed method is, to the best of the knowledge of the author, a novel and will bring new information to the field of mobile OSA screening. The proposed method got good results, with a great potential for improvements, while also shinning light on some important issues related to automatic *at-home* OSA screening.

1.5 Structure of Thesis

The thesis consists of nine chapters, that combined documents the work done during the master's thesis. After this introduction, Chapter 2 will provide the reader with the necessary clinical background knowledge in order to understand the methods used in the thesis. Chapter 3 presents the state of the art literature in the field, and the choice of approach based on it. Chapter 4 is about the steps taken from the idea of creating the method, to actual patient trials. Chapter 5 explains every aspect of the signal acquisition design. Chapter 6 contains the methods used when creating the OSA screening algorithm. Chapter 7 presents the results obtained when analyzing the acquired database. Chapter 8 discuss the results acquired, stating strengths, weaknesses and possible improvements of the system. Finally Chapter 9 evaluates the success of reaching the before mentioned project goals.

2 Clinical Background

Obstructive sleep apnea (OSA) is one of three *sleep disorders*; OSA, Central Sleep Apnea (CSA) and Mixed Sleep Apnea (MSA). This chapter is meant as an introduction to the necessary principles and terminology of Obstructive Sleep Apnea (OSA), in order to comprehend the methods used in the proposed solution, to understand arguments behind each choice. Three aspects of OSA will be included in this section; epidemiology, pathofysiologi and diagnostics of the disease. The focus will be on the diagnostics since it correlates with the project goals, mentioned in the previous chapter.

2.1 Epidemiology

Obstructive sleep apnea is estimated to affect between 2 % and 4 % of the adult population, with a variation between ethnics groups [1]. The biggest risk factor is obesity, which is seen in countries struggling with obesity having an increased prevalence of OSA. Other risks associated with the diseases are wge, gender and smoking [2].

Untreated OSA can result in different complications such as an increased risk of cardivascular diseases such as *chronic heart disease*, *arrhytmias*, *stroke*. Furthermore, other associations are an increased chance of being in an accident, decreased learning ability, etc. due to sleepiness. A study was conducted by Jennum P. et Kjellberg J. [4], where they investigated health, social and economical consequences of diseases in the category of *sleep-disordered breathing*. The result showed a correlation between having *sleep apnea* (from which OSA is a sup-type), and an increased health-related contact as well as increased unemployment rate and socioeconomic cost. These findings together with the prevalence of the disorder, causes a heavy economic load on the society. Additionally, the disorder also decreases the quality of life, which makes it even more important to treat people with OSA as early as possible.

2.2 Patopfysiology

Whenever a breath is taken, air is sucked in from the mouth and/or nose, through the *upper airway* region, into the lungs. The mechanisms responsible of OSA is an obstruction in the upper airways, disturbing the airflow from the nose down the lungs. The upper airway region is composed of soft tissue and muscles, but has no bony support making it prone to collapses. Naturally, this depends on the cross-sectional area of the airway, and the locations of big clumps of soft tissue. A narrower airway is more prone to collapse and also increased soft tissue around the pharyngeal region can lead to a higher pressure on the upper airway (see Fig. 2.1 for an illustration) [2]. This is why ethnic groups with narrower airways than others, and people with high *Body Mass Index* (BMI) are more prone to having OSA, as stated in the previous section.

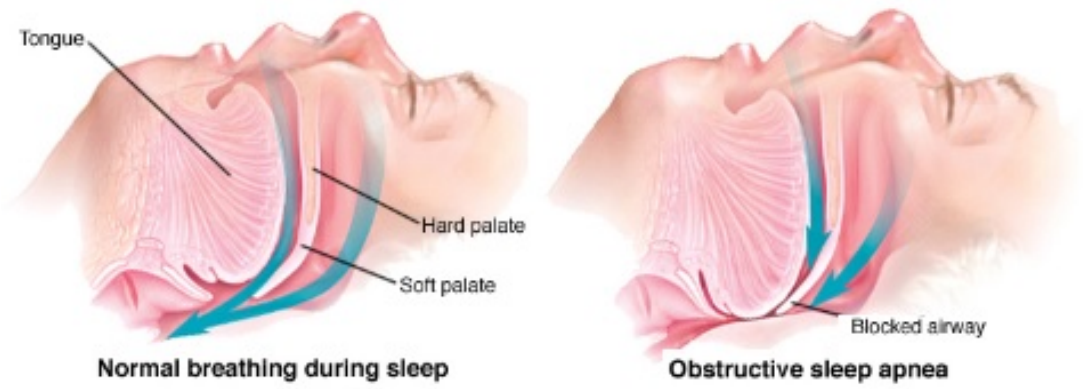


Fig. 2.1: Image of upper airway anatomi during normal breathing (left) and during an obstructive event (right), while sleeping. Notice that in the obstructive event, the tongue can worsen the obstruction (mainly when sleeping in surpine position). From [3].

The reason that patients with OSA normally do not experience any breathing difficulties while awake is that the upper airway muscles are active and makes sure the airway is dilated. This has been validated by studies that showed OSA patients had an increased upper airway muscle activity while awake [2].

2.3 Diagnosing Obstructive Sleep Apnea

Finkel K. J. et al. [6], conducted a study related to the rate of undiagnosed *sleep apnea*, and found out that an estimated 90 % of adults in US suffering from *sleep apnea* are undiagnosed. There are a lot of arguments for this; milder forms of OSA can be symptomless, a lot of the symptoms does not involve pain and therefore less likely to alarm the subject, and so on. The other aspect is that it is a time consuming and costly process for the hospitals to conduct *sleep studies* [20]. Therefore medical professionals are on the lookout for simpler solutions, and have currently defined four cathegories for OSA diagnosing methods, listed below [6].

- I: Classic PSG.
- II: Non-supervised PSG with at least seven biological parameters.
- III: Portable system recording at least four parameters.
- IV: Portable system using only one or two parameters.

2.3.1 Polysomnography

The most traditional way of diagnosing for sleep disorders, including OSA, is the polysomnography (PSG). This is a comprehensive analysis with a wide range of clinical parameters being measured and are thus mostly conducted in a sleep clinic where a medical professional can attach the sensors [16]. In total 8 different sensors are included in the PSG *sleep study*; respiration belts, Snore sensor, Nasal pressure transducer, Electrocardigraph (ECG), Electromyograph (EMG), Electroencephalograph (EEG), oximeter and an Electrooculography (EOG).

All of these sensors are used to diagnose for *sleep apnea*, by looking for *apneic events* during a full night's sleep. Three types of events are noted; *Apneic event*, *Hypapneic event* and *Central*

event, where the latter is not related to OSA. Generally, an obstructive event is occurring when the nasal pressure flow signal is reduced by atleast 90 % with corresponding respiratory effort for over 10 seconds. A hypapneic event is noted when the nasal pressure flow is reduced by more than 30 % in more than 10 seconds if there is a corresponding decrease in oxygen saturation of 4 %. Both types of events require an arousal to be observed in the EEG before they are notated. See Fig. 2.2 for a representation of an *event*.

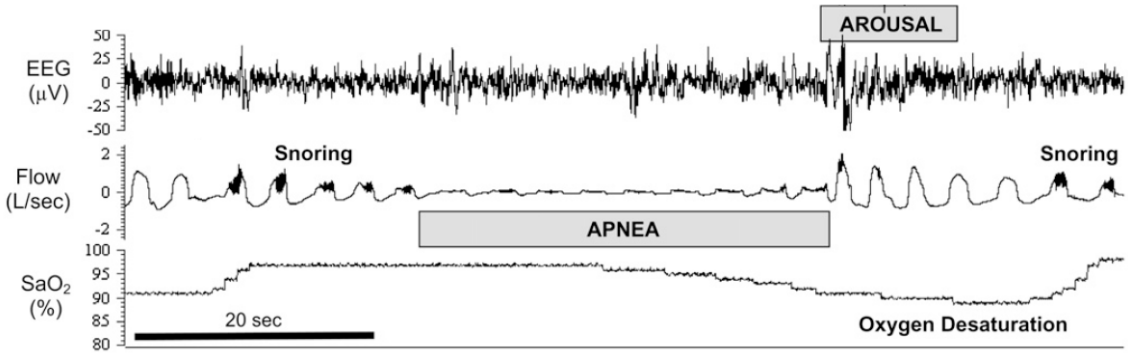


Fig. 2.2: Illustration of an apneic event, by three sensors; EEG (top), Nasal pressure transducer (middle) and oximeter (bottom). Notice the reduced airflow during the event, followed by an arousal in the EEG (subject wakes up) and a decrease in blood oxygen saturation. From [2].

The recordings of the whole night are labeled by medical professionals, using the manual *"Rules for scoring respiratory events in Sleep Apnea"* by the American Acadamy of Sleep Medicine (AASM) [13]. Finally the Apnea Hypapnea Index (AHI) is calculated using Eq. 2.1:

$$\text{AHI} = \frac{\text{Apnea}_{\text{events}} + \text{Hypapneic}_{\text{events}}}{\text{PSG}_{\text{length,h}}} \tag{2.1}$$

, where $\text{Apnea}_{\text{events}}$ is the number of *apneic* events, $\text{Hypapneic}_{\text{events}}$ is number of *hypapneic* events and $\text{PSG}_{\text{length,h}}$ is the length of the recording in hours. AHI is a parameter used for indicating severity of *sleep apnea*. Since it is normal to have a few events during sleep, the AHI threshold which distinguishes between healthy and OSA suffering subjects is in the literature stated as an index value of 5 in the literature [10][15][25]. See Table 2.1, for the different AHI thresholds used in the literature to classify the severity of OSA.

Class	**AHI index**
Healthy	AHI < 5
Mild	$5 =$ AHI < 15
Moderate	$15 =$ AHI < 30
Severe	AHI > 30

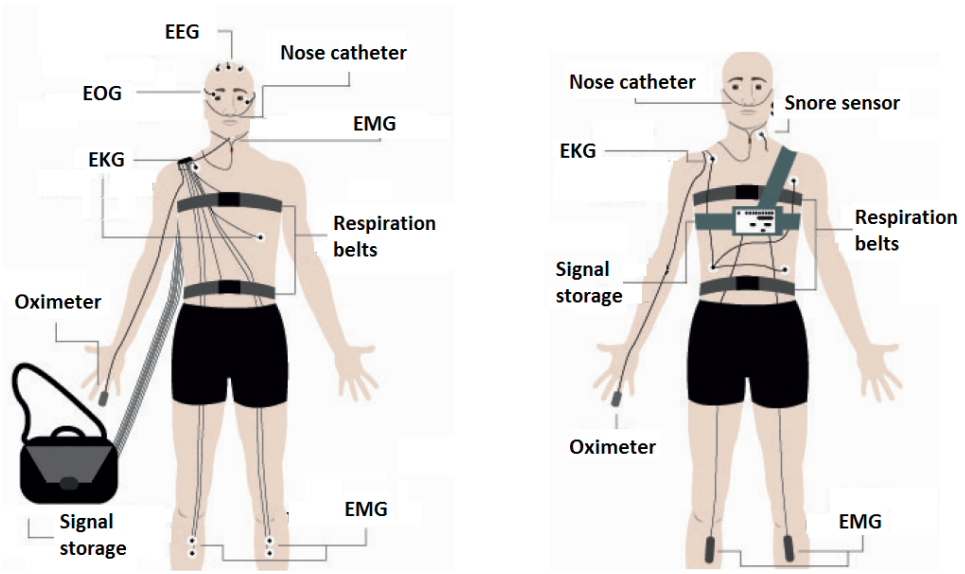
Table 2.1: The AHI thresholds distinguishing between healthy subjects and subjects suffering from OSA, in different degrees, according to the literature.

Even though the literature defined the AHI threshold between healthy and OSA suffering subjects as a AHI value of 5, project superviser dr. med. Poul Jennum, argued that most doctors found

it to be too low, and that an AHI of 10 is more fitting.

2.3.2 Cardiorespiratory Monitoring

The Cardiorespiratory Monitoring (CRM) is a less complex diagnosing tool for Sleep Apnea, and is currently used in practice at Rigshospitalet Glostrup, and is widely considered the standard practice for diagnosing OSA. The CRM uses a total of 6 types of sensors; Nose catheter, Snore sensor, ECG, respiratory belts, oximeter and EMG. Both the experimental setup of the PSG and CRM are shown in Fig. 2.3:



(a) Polysomnography (PSG) **(b)** Cardiorespiratory Monitoring (CRM)

Fig. 2.3: Figure showing the two frequently used methods for acquirering biological signals, for the purpose of diagnosing *sleep apnea*. Fig. 2.3(a) from [11], and Fig. 2.3(b) from [12]. Modified to be in English.

2.3.3 Questionaire

Many have tried to find a way to implement the risk factors of OSA into a method which can assess the severity of the disease, namely questionnaires. These questionnaires are ways of scoring the patients according to the risk factors related to the disease. There are in general three questionnaires used to screen for OSA; STOP-BANG, Berlin and Preoperative Questionnaire and G.A.S.P. The most complex of the three are the Berlin Questionnaire which has 10 questions covering different areas related to sleep apnea. For each question a threshold is defined to make the answers binary, *"yes"* or *"no"*. [17].

3 State of the Art

The scope of this chapter is to get familiar with the current possibilities and developments in the field of *mobile sleep apnea detection*. The knowledge acquired from the literature study will be used as guidelines for choosing the approach, while also ensuring that it will contribute with new knowledge to the field of sleep studies.

The main focus of the literature study are methods involving smartphones for sleep apnea detection. The most relevant set of criteria defined by the author after consultation with his supervisors and in accordance with the project goals, will be reviewed in a standard form; Purpose, Subjects, Equipment, Method, Results and Remarks. The method used in the litterature study can be seen in Appendix B.1.

Finally, a sub-study was created as a result of limited amount of mobile solutions found in the literature. The findings of the sub-study can be seen in section 3.2, and the method used to gather the information, can be seen in Appendix B.2.

3.1 Mobile Sleep Apnea Screening Methods

Monitoring Sound To Quantify Snoring and Sleep Apnea Severy Using a Smartphone: Proof of Concept [18]

Author: Hioshi Nakano, Ph.D.;Kenji Hirayama, Ph.D. Yumiko Sadamisu, R.N.; Ayaka Toshimitsu, M.T.; Hisayuki Fujita, M.T; Shizue Shin, M.T.; Takeshi Tanigawa, Ph.D.

Purpose: To develop a snoring sound monitor by only using a smartphone and its build-in microphone to acquire biological signals, with the aim of quantifying snoring and OSA severity of a subject.

Subjects: A database of 50 subjects who underwent diagnostic PSG due to suspected sleep apnea.

Equipment: A smartphone (SH-12C, Sharp Corp.) running android OS (v2.33), and the build-in microphone.

Methods: A smartphone was attached to the anterior chest wall, with the microphone turning towards the neck. Using a custom application, the system acquired sound data in 0.1 second windows (11025Hz sampling frequency, with hanning window) and calculated power spectra using fast fourier transform (FFT). This process was repeated every 0.2 seconds.

Three different analysis was conducted on the sound signal; intensity, snoring sounds and respiratory events. The snoring sound intensity was determine by calibrating the program followed

by calculations of 1 percentile sound-pressure levels (L_1) and the equivalent sound pressure level (L_{eq}). Features for detecting snoring was done on the basis of a spectral analysis of 10 subjects, from which snoring segments were extracted. To detect respiratory events, sound power (50-2000Hz) time series data were generated and lowpass filtered (f_{cutoff} 0.05 Hz) to detect so-called *sound dips*. Thresholds for detecting sound dips was based on 10 subjects from the database. See Fig. 3.1 for a schematic illustration of proposed method.

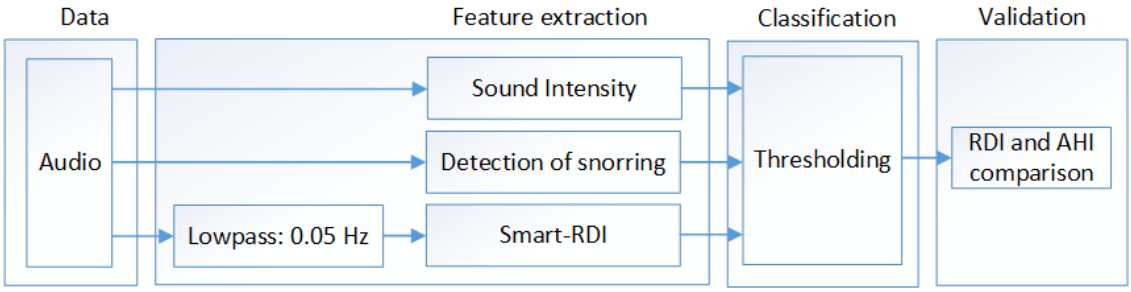


Fig. 3.1: Illustration of the implemented method by Nakano et al. [18].

The correlation coefficient of L_1 between the smartphone system and the PSG was r = 0.75 for the tracheal sound and r = 0.92 for ambient sound. The correlation coefficient of L_{eq} between the smartphone system and the PSG was r = 0.72 for the tracheal sound and r = 0.82 for ambient sound. The sensitivity ranged from 77% to 92% and the specificity from 82% to 96%, depending on the AHI threshold.

Remarks

The study proved that there was some correlation in snoring parameters between their system and the PSG, and the classification results was decent. However, the data was from a clinical database of PSG recordings. Since they expressed concern towards the algorithms being prone to noise, it might not be fit for an *at-home screening system*.

Classifying obstructive sleep apnea using smartphones [19]

Author: Mamoun Al-Mardini, Fadi Aloul, Assim Sagahyroon, Luai Al-Husseini

Purpose: A system capable of *at-home* biological signal acquisition and analysis to detect the presence and severity of OSA.

Subjects: 15 subjects (14 male, 1 female) from a clinical database. 7 were healthy, and 8 suffered from OSA.

Equipment: An oximeter, a microphone, a accelerometer and a smartphone.

Methods: SpO2 is measured with a bluetooth oximeter placed at the fingertip or earlobe, the microphone is placed on the throat, and the smartphone with the accelerometer is placed on the arm around the biceps.

Apneic events is detected with the oximeter by calculating the avere axygen saturation in one second windows. If it exceeded a certain threshold for 10 consecutive seconds, an event were detected. The threshold was calculated as an average *SpO2* in the first few minutes. From these events an, Oxygen desaturation idex (ODI) was calculated, as the rate of events pr. hour. To remove noise *SpO2* readings, Signal vector magnitude was calculated from the accelerometer signal, and if it exceeded a threshold, the *SpO2* readings from the corresponding time-segment would be discarded. Respiratory effort is found by first extracting frequencies between 200 and 800 Hz from the audio data, followed by the same event-finding procedure used for the oximeter measurements. Finally, the AHI index (events/hour) were calculated and divided with the ODI. The final diagnostic is based on the average of the ODI and AHI score. See Fig. 3.2 for a schematic illustration of the proposed method.

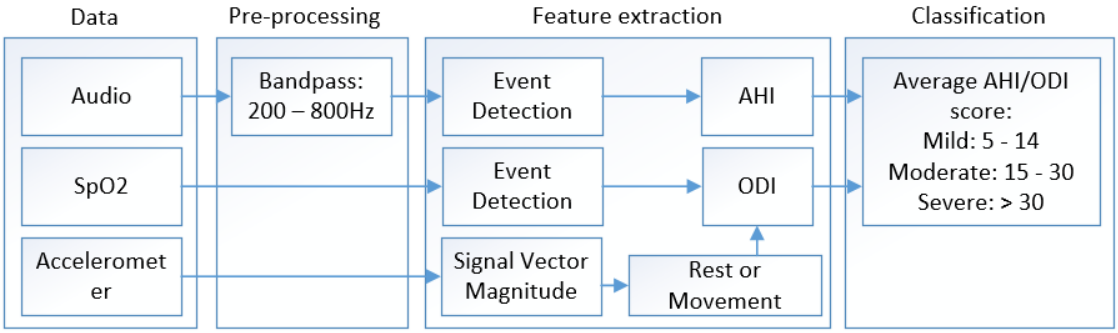


Fig. 3.2: Illustration of the implemented method, Al-Mardini M. et al. [19].

Results: Results showed that the AHI index proved to be close to the gold standard, diagnosing all 15 subjects correctly as being healthy or suffering form OSA. Furthermore, of the subjects with suffering from OSA, 7 out of 8 was correctly classified according to 3 severity severity classes; mild, moderate and severe (accuracy of 87.5 %).

Remarks

Even though this study achieved the best performance of the methods found in the literature search, it was based on relatively few subject PSG recordings from a clinical database. The signal processing algorithm had some interesting aspects, and might be improved to make it more robust to noise.

SleepAp: An Automated Obstructive Sleep Apnoea Screening Applicaion for Smartphones [20]

Author: Joachim Behar, Aloife Roebuck, Mohammed Shahid, Jonathan Daly, Andre Hallack, Niclas Palmius, John Stradling and Gari D. Clifford, senior member, IEEE

Purpose: To create a simple *at-home* screening device, and use novel machine learning approaches to perform diagnostics related to OSA.

Subjects: Clinical database of 856 subjects, where 735 is used for training and 121 are used for testing.

Equipment: Microphone, accelerometer, oximeter and smartphone.

Methods: An audio signal was recorded from a microphone placed next to the nose on line with the nostrils, accelerations was gathered from a build-in smartphone accelerometer placed on the arm, and an *SpO2* readings from an oximeter placed on the finger. Demographic information was collected using a STOP-BANG questionnaire. The actigraphy was sampled with 4 Hz and bandpass filtered afterwards using a FIR filter with $\text{freq}_{\text{cutoff}}$ at 0.75-1.65 Hz. After this, the variance in all dimensions was calculated over 1 second intervals, followed by MSE calculations. The accelerometer were also used to determine the body position. This was recorded with a sampling frequency of 4 Hz, to calculate total time spent in each position during the night. The audio signal was sampled with 8 kHz in a single-channel (mono) 16-bit pulse-code calculation. The acquired audio signal was processed in a semilar way as the accelerometer data. The PPG was recorded using a oximeter placed on the finger, with a sampling frequency of 75 Hz. ODI was defined as a decrease in oxygen saturation of at least 4 % compared to the average. Finally, features are processed using a SVM in different feature combinations. See Fig. 3.3 for schematic illustration.

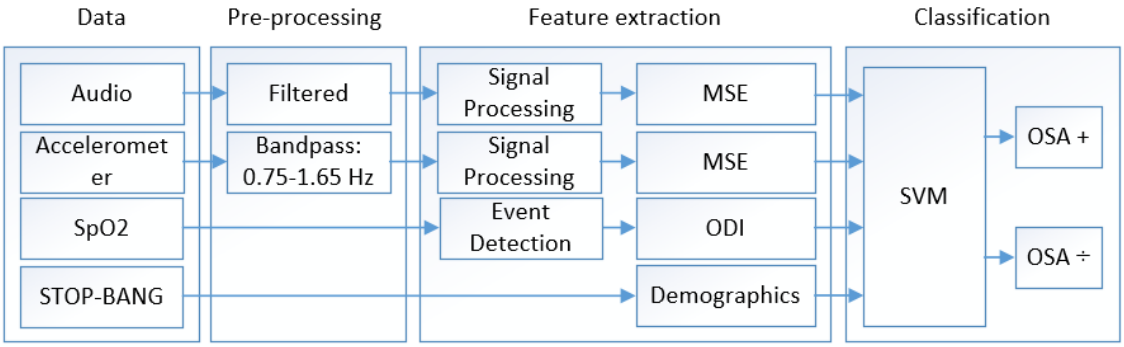


Fig. 3.3: Illustration of the implemented method by Behar J. et al. [20].

Results: All sorts of feature combinations were tested with the SVM. The best results from the first and second experiment was achieved by using a combination of audio, actigraphy and ODI, resulting in and accuracy of 88.4 % and 92.3 % respectively. The solution performed poorly with mild OSA subjects, with 3/4 classified as being healthy.

Remarks

The study proved to be able to classify subjects with a relatively good accuracy, but it only classified into two classes; OSA and non-OSA, where subjects suffering from mild OSA got labeled as being healthy 3 out of 4 times.

HealthGear: Automatic Sleep Apnoea Detection and Monitoring with a Mobile Phone [21]

Author: Nuria Oliver and Fernando Flores-Mangas

Purpose: Create a real-time wearable system for monitoring, visualizing and analyzing physiological signals for assessing sleep apnea severity.

Subjects: 20 subjects (80 % male) in the age between 25 and 65. Of those, 30 % were healthy, and the rest had OSA or were suspected of having it.

Equipment: Nonin's Flex Oximeter and an Autodiovoc SMT5600 GSM cell phone.

Methods: The oximeter was placed on the subjects earlobe or finger and measurements were send via bluetooth to the cell phone.

Method 1: An event starts as soon as the lower baseline oxygen saturation is met, and has no minimum duration. This baseline was found by calculating the moving average over 5 minutes only using the top 5 % of samples. Different saturation levels were defined (below 5, 10 and 15 % of baseline oxygen saturation level), and severity was calculated based on length and saturation level of events. Finally, a histogram of time spent in each SpO2-level were made.

Method 2: A periodogram of the mean-subtracted oximetry signal was computed, using a sliding window of 834 samples. The data was weighted using a using a Hamming-window, implemented with a 1024-point FFT. The authors noted a peak in the frequency range of 0.015 - 0.04 Hz, which they used to detect sleep apnoea events. They also concluded that a larger amplitude of a peak corresponded to a more severe event. See Fig. 3.4 for schematic illustration of both methods.

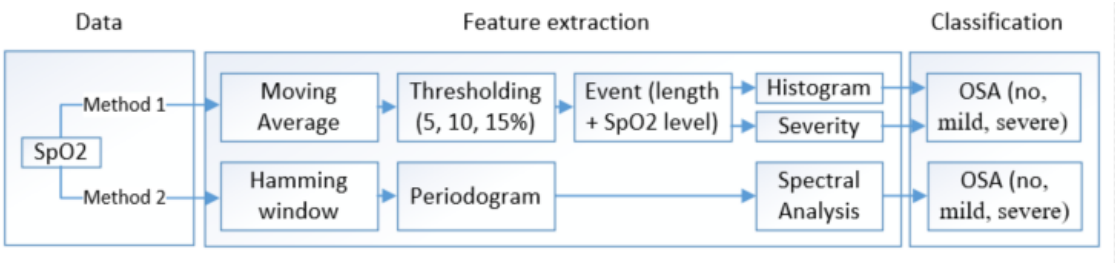


Fig. 3.4: Illustration of the implemented method by Oliver N. et al. [21].

Results: The algorithm succeeded in detecting all 3 cases of known OSA. Unfortunately, they were not able to run a PSG study simultaneously as they tested the HealthGear, so it could not be validated (specificity and sensitivity).

Remarks

The study stood out by only using one type of signal for the analysis, which made it very interesting. However, since it was not validated it is hard to conclude if its usable.

Sleep Apnea Monitoring Using Mobile Phones [22]

Author: Shamma Alqassim, Madhumeta Ganesh, Shaheen Khoja, Meher Zaidi, Fadi Aloul, Assim Sagahyroon

Purpose: A mobile sleep apnea screening tool, only using build in sensors from a smartphone. The aim is be able to distinguish not only between SOA and non-OSA subjects, but also stage the severity into three levels; mild, moderate and severe.

Subjects: None.

Equipment: Microphone and accelerometer, both build-in smartphone (Sony Xperia neo V and HTC 7 are used).

Methods: Both the audio and accelerometer data were recorded simultaneously, and send wirelessly to a computer with MATLAB. Frequency spectrum of the audio data was found by using short-time Fourier transform, and placed in an array. Percentage of sleep time the patient was snoring, was found by calculating time spent in frequencies over a given threshold (800 Hz). The accelerometer data was processed with the root-sum-square (RSS) method. This methods calculates the overall changes in acceleration. The unit G (unit in acceleration) was used, where G = 1 was no movement, and G = 1.03 was a threshold for when a considerate amount of movement occurred, considered to be related to an event. The final sleep apnoea score was found by taking a weighting percentage of the time spent above the movement threshold and apnoea snoring minutes. See Fig. 3.5 for schematic illustration of both methods.

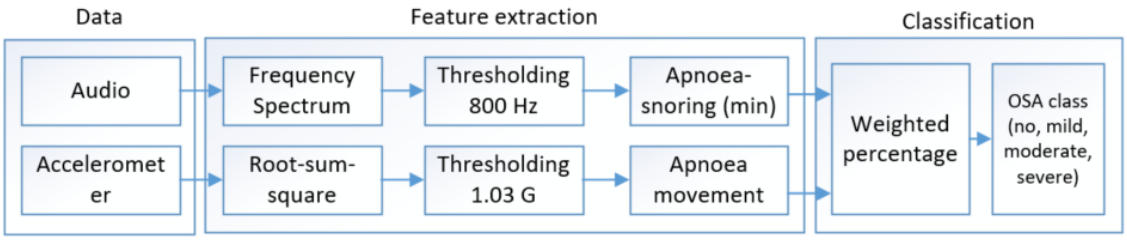


Fig. 3.5: Illustration of the implemented method by Alwassim S. et al. [22].

Results: The system was only tested for functionality, not on any subjects, and therefore no results has been published.

Remarks

Even though it was not validated, it did propose some good ideas regarding processing of accelerometer and audio data. Especially the idea of relating movement to *apneic* events was interesting.

Apnea MedAssist II: A smart phone based system for sleep apnea assessment [25]

Author: Cheng Shi, Mehrdad Nourani, Gopal Gupta and Lakshman Tamil

Purpose: A mobile system, for screening subjects for OSA and classifying not only if they suffer from OSA, but also diagnose the severity into three categories; mild, moderate or severe.

Subjects: Database of 34 sleep studies, annotated by experts, containing both both ECG and SpO2 signals.

Equipment: ECG, oximeter and two breath belts for signal recording connected to a smartphone (Samsung Galaxy S3) for signal processing and classifications.

Methods: The ECG is placed as usual on the chest, the oximeter on the finger and the breath belts around the chest and abdomen. Signals are recorded and sent to the smartphone, where the app MedAssist handles the data process. The ECG was sampled with a sampling frequency of 250 Hz. Data segments (one minute length) was passed through an automatic wavelet filter designed to remove noise and extract useful information. An adaptive threshold was used for each wavelet level to further clear the QRS segments from the P and T spikes. The Heart Rate Variability (HRV) and morphology change (EDR) was then calculated from the ECG data, and a total of 10 features was selected to be used in the classification process.

The breath signal was also preprocessed with a wavelet transform equal to the one used on the ECG signal, and the final breathing wave was converted to a rectangular signal for more efficient feature computation. The torso respiration is found with the two breathing sensors with; amplitude (proportional to the lung volume) and frequency (proportional to breath rate). Since OSA events can cause amplitude, frequency and phase change in the wave, five different features are extracted from it. Five features were extracted from the raw SpO2 signal, and directly used for the classification. Three different SVM classifiers was used, - one for each sensor, with K-cross validation used to prevent over-fitting. Finally the AHI index was calculated and OSA severity was classified; AHI of ł 5 is healthy, 5-15 = mild, 16-30 = moderate and AHI > 30 = severe. To find the optimal feature set, Correlation Feature Selection (CFS) was used. See Fig. 3.6 for schematic illustration (Optimal features listed).

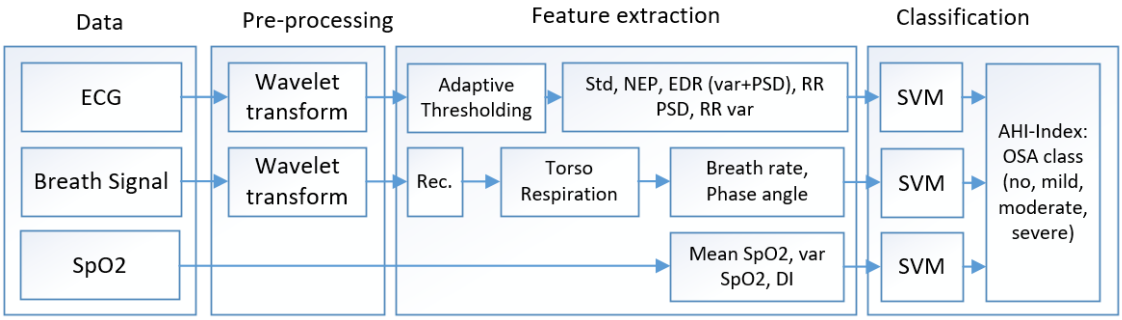


Fig. 3.6: Illustration of the implemented method by Shi C. et al. [25].

Results: MedAssistII proved to achieve great results when used on the clinical database of 35 subjects, with Specificities ranging from 87 - 91 %, sensitivities of 85 - 88 % and Accuracies from 89 - 98 %.

Remarks

The study did indeed receive good results, but the best feature set showed that the ECG provided the best information.

3.2 Sub-Study: non Mobile Methods

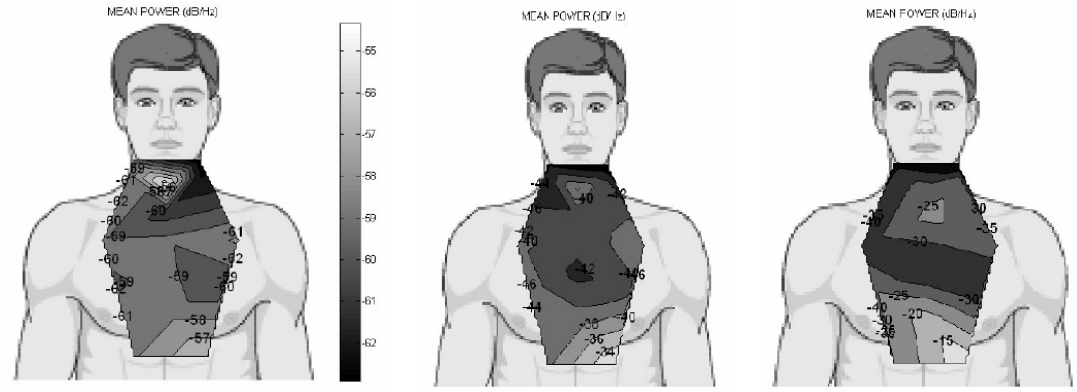
Due to the very limited amount of mobile solutions for sleep apnea assessment, a sub-study was conducted, to further look for potential methods, with a focus on feature extraction and classification.

Audio Data Acquisition

Halevi M. et al. [28], had some interesting findings, related to distinguishing between three events; normal breath, apnea, hypopnea. Especially the *SuperSnore* estimation, which is a measurement of spectral power distribution between the pre-event and post-event times. Solá-Soler J et al. [27] analyzed snoring segments via a set of features derived from the Power Spectral Density of segments, to distinguish snoring segments from normal respiration and no breath segments.

Accelerometer Data Acquisition

Rendón D.B et al. [23] investigated for body vibrations related to different biological events. This pointed at the fact that placing the accelerometer elsewhere could also provide important biological measures known to be related to obstructive sleep apnea; respiration pattern, ECG and snoring [23]. Furthermore, the author is also in the belief that the parameters *body movement* and *sleep position*, can still be extracted from this location. Below in Fig. 3.7, is seen the outcome of their study in three subfigures.



(a) Mapping of the Neck and Thorax: Snore

(b) Mapping of the Neck and Thorax: Heart Rate

(c) Mapping of the Neck and Thorax: Breathing

Fig. 3.7: The outcome of the study by Rendón D. B. et al [23]. The figures shows the mean power (dB) and the lighter the area the higher dB.

Since *snoring* is much easier to detect from the microphone than breathing and heart rate, it was considered a less important parameter to extract from the accelerometer measurements. Therefore two different accelerometer locations came out of this; placement on the *suprasternal notch* or *intersection of the lower costal margin and the midclavicular line.* Morillo D.S. et al [24] also investigated for accelerometer placement for OSA indications, and their findings backs up Rendón D.B et al. [23] of an optimal accelerometer placement being on the *suprasternal notch.*

3.3 Summary and Choice of Approach

The process of reviewing the current literature in the field of OSA screening using mobile solutions, proved that even though many different approaches were used, some trends were seen.

When it comes to data acquisition, mainly three sensors were used, but in different combinations; *Microphone*, *Accelerometer* and *Oximeter*. All of the studies that used *SpO2* measurements collected the data by placing an oximeter on the finger or earlobe. The accelerometer measurements was in all cases acquired by placing a phone, capable of measuring accelerations, on the arm fixed by using some sort of armband. The audio acquisition was done in three different locations; throat, cheek or ambient (placed distant from the patient).

The preprocessing steps for the accelerometer and audio signal was always done by using some sort of bandpass filter, opposite to the oximeter data which was always used raw. Behar J. et al. [20] bandpassed the accelerometer signal with cutoff frequencies of 0.75 and 1.65, arguing that these are the frequencies related to breathing. Both Alqassim S. [22] and Al-Mardini M. et al. [19] extracted features from the raw signal, since they determined movement using a combined value of all the axis, making the gravitational influence on a specific axis irrelevant. Al-Mardini et al. [19], used a bandpass filter with cutoff frequencies of 200 Hz and 800 Hz to detect apneic events. Nakano H. et al. tried to detect events by lowpass filtering the data with a cutoff frequency of 0.5 Hz, and then looked for "power dips", represented by segments with lower signal power.

A broad range of feature extraction methods was used in the state of the art articles. Al-Mardini et al. [19] and Behar J. et al [20] both used the *oximeter* for event detection using slightly different event definitions, to calculate ODI. Oliver N. et al [21] extended this method by dividing the events into multiple stages depending on the desaturation. Chi S. et al [25] tried a whole different approach and looked for trends in the signal rather than specific events. The accelerometer measurements was analyzed very differently between the state of the art approaches. Al-Mardini et al. [19] only used it indirectly for detecting movement in order to determine when the oximeter measurements was variable or not. Behar J. et al. [20] and Alqassim S. et al [22] used it more directly by looking for movement related to sleep apnea movement. The *Audio* signal was used in many ways. When the microphone was placed on the throat it was to focus on the snoring, which is a typical (but not necessarily) sign of sleep apnea. Behar J. et al [20] used this method to look locate apneic events, by looking at a decrease in audio signal. Others, such as Alwassim S. et al. [22] and Nakano H. et al [18] used the audio signal to calculate time spent snoring.

Three different classification methods were seen used; thresholding, weighted percentage and SVM, to place the subjects into two or four classes. The former two are simple methods that, like the AHI, classifies subjects into categories by placing thresholds. The SVM is a complex binary machine learning method used by Behar J. et al. [20]. Shi C. et al. [25] extended the classification to a 4-class problem also using an SVM.

3.3.1 Choice of Approach

The choice of approach was based on the knowledge gained from the state of the art articles reviewed in the previous section. The key points of each approach are listed in Table. 3.1.

Study	Subjects	Sensors	Method	Results
Nakano H. et al. 2014	50 PSG subjects	Smartphone, build-in Mic.	SI, snorring, smart -RDI, thresholding	Sens = 87 % Spec = 82 % Acc = NA
Al-Mardini. et al. 2014	15 subjects	Microphone Oximeter accelerometer	Bandpass-filter, ODI, AHI ODI/AHI average	Sens = 100 % Spec = NA Acc = 87.5 %
Behar J. et al. 2015	735 subjects, 50% with OSA	Microphone Oximeter accelerometer STOP-BANG	Bandpass, MSE, ODI, demographics, SVM	Sens = NA Spec = NA Acc = 88-92 %
Oliver N. et al. 2007	20 subjects, 80 % male	Oximeter	M1: Mov.avg. Thres-, hold, Histogram, SEV M2: HW, periodogram Spectral Analysis	Sens = NA Spec = NA Acc = NA
Alwassim S. et al. 2012	NA	Microphone Audio	FreqSpec., Thres-hold, RSS, event-detection	Sens = NA Spec = NA Acc = NA
Shi C. et al. 2013	Database of 34 subjects	ECG Oximeter Breath Signal	Wavelet, Adapt.-thresholding, SVM 3	Sens = 85-88 % Spec = 87-91 % Acc = 89-98 %

Table 3.1: Overview of the different state of the art solutions regarding mobile assessment of OSA. Only the most important aspects of each study are listed, in order to give a simple overview of similarities and differences between them. NA = Not available, meaning that it was not specified in the article.

Sensors Included in Experimental Design

The sensors that will be used for data acquisition in this project will be the build-in accelerometer of a smartphone and a microphone. Even though the literature proved that the *SpO2* index was a popular parameter to measure, it is not included in this project. It was determined on the basis of the project goals of creating a screening solution, most people are able to conduct themselves at home. On this basis, the *oximeter* was discarded as a choice of sensor.

Sensor Placement

On the basis of the literature (mainly from sub-study) and discussions with project supervisor dr. med. Poul Jennum, the accelerometer placement was chosen to be on the suprasternum, something not yet seen in mobile apnea detection solutions. The author believes that this could be beneficial and bring new knowledge to the field.

The locations for sound acquisition was chosen to be on the cheek next to the nose, based on the literature study, medical personnel at Glostrup Rigshospital and discussions with project supervisor dr. med. Poul Jennum. This allows the airflow of the subject to be recorded which is an important signal when looking for *apnea* and *hypapnea* events [42].

Sample Frequencies

Since Android SmartPhones do not ensure high temporal precision, the sampling rate was set as high as possible (50 Hz). The audio sampling frequency was set to 16 kHz, even though the literature stated that frequencies related to breathing and snoring is below 2 kHz [18] [27]. This was due to the fact, that future applications of the data could involve speech analysis, which requires high sampling rates of the audio signals.

Feature Extraction

The scope of the accelerometer feature extraction process is two things; investigate the quality of *respiration* (RR) and *heart rate* (HR) extracted from the signal, and finding the optimal features for the classification process. If either RR or HR estimations proves to be good, they might be considered to be used in the classification process. Features extracted from the accelerometer signal will *body position*, *apneic movement* [22]. The body position feature will be represented as percentage of the night spent in *surpine* sleep position, since this position is highly correlated with increased OSA events. The choice of approach regarding audio features extraction will be based on the idea of event detection as Al-Mardini M. et al. [19] used, it might be necessary to modify it to make it more robust to noise.

Classifier and Validation

Due to multiple classification problems had to be solved, multiple classifiers was tested. SVM was the classifier of choice for the 2-class problems, because it is known as a great binary classifier [37] and was used in multiple state of the art studies [20][25]. However, since the SVM inherently is a binary classifier, multi-class problems requires multiple SVM's to be trained, which introduces a set of problems, and therefore another classifier must be chosen to address the multi-class problem. The second classifier used, is chosen to be a Bagged Decision Tree classifier, since it is simple classifier, known to perform well in multi-class problems with small dataset and small amount of features [37].

3.4 Experimental Equipment

Choice of equipment is an important step in setting up an experiment. This section is used to review the current technologies available, and based on this, choose equipment fit for the experiment.

3.4.1 Microphone Choice

The studies investigated in the previous section, lagged microphone specifications. However, the frequencies used in the studies where in the interval of 0.1 kHz - 5 kHz, which should be kept in mind. To narrow the search for potential microphones, some requirements were defined seen below:

- Frequency response had to include the interval 0.1 kHz - 5 kHz
- The size of the microphone should be as small of possible, and weigh no more than 50 g
- Compatible with Android smartphones (mini-jack).

Below in Table 3.2 is seen the outcome of the search with the given requirements.

Product	**Frequency Response**	**dB-scale**	**SNR**	**Impedance**	**Weight**
Somnomedics Mic	40 Hz - 5 kHz				
TP-WM07	50 Hz - 16 kHz		60 dB	680 Ω	30 g
Olsen Mic	50 Hz - 16 kHz	-60 dB ± 3 dB			
Speedlink Sl - 8691-SBK-01	100 Hz - 15 kHz				
Azden Ex-503	30 Hz - 18 kHz	-44 dB		2.2 kΩ	3 g
Boya BY -LM10	65 Hz - 18 kHz	-3 dB ± 3 dB	74 dB		17 g
Audio-Tech -nica ATR3350	50 Hz - 18 kHz	-54 dB		1 kΩ	6 g
Rode SmartLav+	20 Hz - 20 kHz	-35 dB	115 dB		6 g
Senheiser ME 2	30 Hz - 20 kHz		130 dB		

Table 3.2: Overview of potential microphones. Blank spaces, indicates information was not provided by the manufacturer.

Since the microphone in the CRM (Somnomedics) is only used to detect snoring sounds, it has a very low frequency response. Since the microphone will be worn on the cheek or throat the size, shape and weight was important in the decision process. The *Audio-Technica ATR3350* was one of the lightest microphones and the investigation led to the knowledge that it had its own battery. The advantage of this is that it saves battery from the smartphone, however, the disadvantage of further adding a potential source of error (low battery or forgetting to turn it on) ruled this microphone out.

Rode SmartLav+ proved to have very promising specifications, since it has a very broad frequency response, weights only 6g and the only specification where it is succeeded in, is SNR. Furthermore, it received good reviews, and was used frequently in professionally both by journalists and musicians. The reason that such a high quality microphone is chosen, is to make the acquired data more widely usable in future studies, for example related to speech during sleep.

3.4.2 Choice of Mobile Device

To be able to find the best choice of mobile device to conduct the experiment, it is necessary to define some particular important specifications related to the success of the project. Since the user will be wearing and sleeping with the device a whole night, the smartphone dimensions have to be considered. Furthermore, the battery must be able to hold for a whole night of data gathering. Finally, the sampling rates of the physiological signals are also important to have in mind. See Table 3.3 for a few specifications of the researched smartphones.

The sampling frequencies of the accelerometer changes between which smartphone is used, and can often be set in four stages; *Normal*, *UI*, *Game* or *Fastest*. However, when the phone is put into sleep mode (screen turned off), the sampling frequencies of most phones goes down.

Product	Dimensions (L-W-D in mm)	Battery time Talking	Acc. fs - Awake (hz)	Acc. fs - Sleep
LG Nexus 5	137.9-69.2-8.6	17 hours	Normal: 15 Game: 50 Fastest: 196	Normal: 4.96 Game: 50 Fastest: 198
LG Nexus S	123.9-63-10.9	6.75 hours	Normal: 49.51 Game: 49.51 Fastest: 49.51	6.06 for all modes
HTC wildfire	106-60.4-12	8 hours	Normal: 4 Game: 22.2 Fastest: 38.40	No sampling
iPhone 6	138.1-67-6.9	14 hours	1-100	No information found

Table 3.3: Overview of potential smartphones. Dimensions and battery time were listed in [29]. The sampling frequencies were stated from users in [30]. No information regarding iPhone 6 sampling rates were found. The battery time while talking was used, since this represents the battery time while the phone is used.

Having the screen turned on for a whole night would drain a lot of power, and changing sampling frequency in the middle of an experiment is not acceptable.

One of the few phones that does not have this problem is the *LG Nexus 5*. When the sampling mode is set to *GAME* it samples with approximately 50 samples/second, both in normal and sleep mode. The dimensions (137.9 x 69.2 x 8.6 mm) is relative big, but not that different from other smartphones having the necessary technical specifications to perform the experiment.

No scientific studies regarding sample rates of different phones excist therefore the information comes from users of 'stackoverflow.com' [30]. Since this is a questionably source, the author validated the results related to the *LG Nexus 5* once it was chosen as the best smartphone for use in this study.

3.4.3 Choice of Fixation Method

The choice of fixation method was done on the basis of a few criteria. First, the armband had to be sweat resistant, so that it could protect the smartphone from the acidic environment of a sleeping person. The second criteria were that it had to have dimensions so that matched a phone, the size of the LG Nexus 5, so that it would not move inside the container. Finally, method had to be comfortable to wear.

The fixation method of choice, was the NUPO armband, since it had all of the above specifications. However, since most commercial smartphone holders, are made to be fixated around the arm, like the NUPO armband. A Velcro strap had to be used to extend the armband so that it could reach around the chest. These straps were the same as the ones used to fixate the CRM when conducting a sleep study.

4 Development of a Medical Device

This chapter explains the steps taken from having the idea of medical device to actual clinical experiments. It includes thoughts gone into the design, documents needed to file for regulatory approvals and the preliminary tests needed in order to prove that the system works, before beginning the clinical trials.

4.1 Designing the Product

The product design had to be simple and easy to use, for any adult to be able to conduct the signal acquisition without any supervision at home. At the same time it was important that the necessary information was gathered, for the data analysis. Thus, the challenge was to achieve balance between creating a simple solution capable of performing satisfactory data acquisition.

The result was a system consisting of two sensors; accelerometer and microphone, and an Android application managing the data acquisition and with a build-in questionnaire (see section 5) for detailed explanation of the system.

4.2 Regulatory Units

In such a controlled environment as the medical sector, it is required to follow a number of formalities, before tests can be conducted on patients. The authorities contacted for this project to be approved was *"The Danish Council of Ethics"*, *The National Board of Health* and *Danish Data Protection Agency*. This was done by a the previous master student, Martin Guul, before the author started working on the project.

4.2.1 Experiment Documentation

To be able to file for approval of the scientific study, the outlines needs to be described and documented in a *Experimental Protocol*. Due to a lot of changes that has been made from the prior project, a new version of the old protocol was written, and can be seen in Appendix C.2. It is important to notify that the *Ethics* section of the protocol is the exact same as the previous protocol written by Martin Guul, since the protocol created by the author is based on the previous one from Martin Guul.

4.3 Preliminary Tests

Before beginning the clinical investigation, the robustness of the system had to be proven. Therefore a number of preliminary tests were conducted prior to the actual data acquisition on patients. The most important aspects of the system that had to be tested were:

- **Quality of Acquired Biomedical Signals**: Did the acquired biomedical signals live up to a satisfactory quality? Is it possible to identify key patterns in the signals such as respiration, snoring, sleep position and so on?
- **Simplicity**: Is the system simple enough for adults of all ages to be able to conduct the analysis?
- **Whole Night Measurements**: Is the battery life-time long enough for a whole nights sleep? Is there enough storage on the phone to store a whole night of signal acquisition?
- **Comparing Performance to CRM**: How will the two system be synchronized in order to compare the performance?

A range of tests were conducted by the author on himself, in order to test the above characteristics. Afterwards, the quality of the acquired biomedical signals, were investigated by the author and his supervisors. When the signal quality of the signals aquired from the author were deemed adequate, the tests were expanded to include family members and friends of the author. This was done in order to test the robustness of the system across multiple test persons of different hight, weight and age. A total of 6 different test subjects were evaluated to ensure that the signal acquisition procedure was usable by people of different body conformations and age.

The plots used to prove the that the proposed system had potential can be found in Appendix D.1.

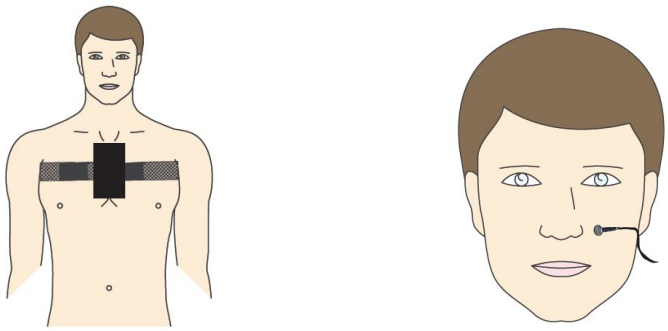
5 Biomedical Signal Acquisition

This chapter describes the different tools and procedures used to acquire the data. It is explained in three sections; *Experimental Design*, *MAS Android Application* and *Experimental Procedure*.

The first section introduces the different methods of gathering data, and is inspired by state of the art analysis as well as conversations with the project supervisors. The second section, explains the *Android application* used to read the data measurements and save them into a data structure. The *Android application* used, is a **optimized version** of the application created by previous master student Martin Guul. Changes and optimization made by the author are listed in Appendix G.1. The third and last section describes the *Experimental Procedure* used for each recording, from first contact with the subject until all data had been archived.

5.1 Experimental Design

The experimental design was chosen to be a smartphone inside a modified armband fixated around the chest, so that the phone was located over the *suprasternum*. The location was based on the findings of Rendón D.B. et al. [23], who concluded that this was the best placement for reading of respiration and heart rate estimation. A microphone was also used, placed left to the nose on the line of the nostrils, based on the study by Al-Mardini M. et al. [19]. A graphical illustration of the final test setup can be seen in Fig. 5.1.



(a) Smartphone placement

(b) Microphone placement

Fig. 5.1: The two figures are graphical representations of the experimental design. Notice that the tape used to fixate the microphone is not shown, to make the placement and angle of the microphone more clear.

Accelerometer

The accelerometer used, was the 3-axis MEMS accelerometer inside the LG Nexus 5 smartphone. It was placed on *suprasternum* as high as possible (see Fig. 5.1a), with the screen turning away from the subject and the jack input pointing upwards. This makes it possible to measure accelerations related to breathing patterns, arousals and body position changes during the night. The size of the smartphone makes it important to know the placement of the accelerometer. In the LG Nexus 5, it is placed in the upper right corner of the phone (when looking at the screen). The coordinate system of the tri-axis accelerometer can be seen in Fig. 5.2.

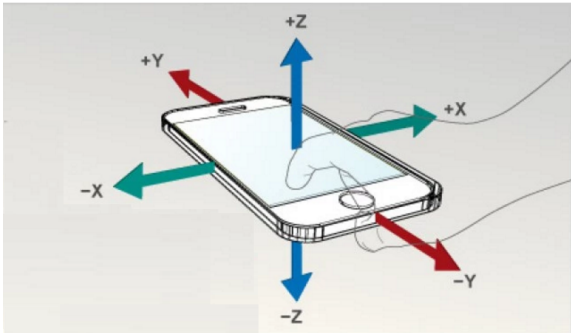


Fig. 5.2: Illustration of the the coordinate system of the build-in accelerometer. The smartphone used in the experimental setup, LG Nexus 5, has the accelerometer placed in the upper right corner (when looking at the screen). From [32].

Since it is essential that the smartphone is kept in place for the whole night, a modified NUPO armband is used. The choice of fixation equipment was based on the current available equipment found on the internet and in stores. The requirements was that it had to be sweat-resistant, fit the size of the LG Nexus 5 so that it did not move around too much inside the case and the company needed to be able to provide a digital check. Therefore the armband both keeps the smartphone in place and also protects the smartphone from the acidous surroundings that can occur if the subject sweats a lot while sleeping.

Audio

The microphone is placed on the cheek on line with the nostrils as seen in Fig. 5.1b. This microphone placement allows the recording of respiratory effort and snoring sounds. Medical tape is used to fixate the microphone.

Questionnaire

The chosen questionnaire for the proposed method was the STOP-BANG, based on the findings of Behar J. et al. [20] and conversation with the project supervisors. The STOP-BANG questionnaire is implemented with modifications in the three first questions, which was originally questions with binary answers, like the rest. This is to be able to provide statistical knowledge about the study participants. The questions asked is listed below.

- What is your gender?

- How old are you?
- What is your BMI?
- Do you snore very loud?
- Do you often feel tired, exhausted or sleepy during the day?
- Have anyone ever observed you stopped breathing during your sleep?
- Do you have high blood pressure or are you currently being treated for it?
- Does you neck-circumference exceeds 40 cm? (shirt neck-size)

5.2 MAS Android Application

A central part of the data acquisition is developing software, capable of accessing the sensors used for the study, sample the continuous physiological signals and store the measurements in data structures. Furthermore, the users also have to be able to fill out a questionnaire inside the application. This was done by creating the android application “MAS”. To see a flowchart of the whole application, see Fig. 5.3.

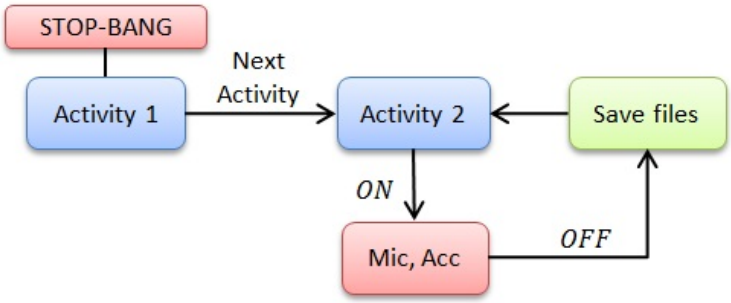
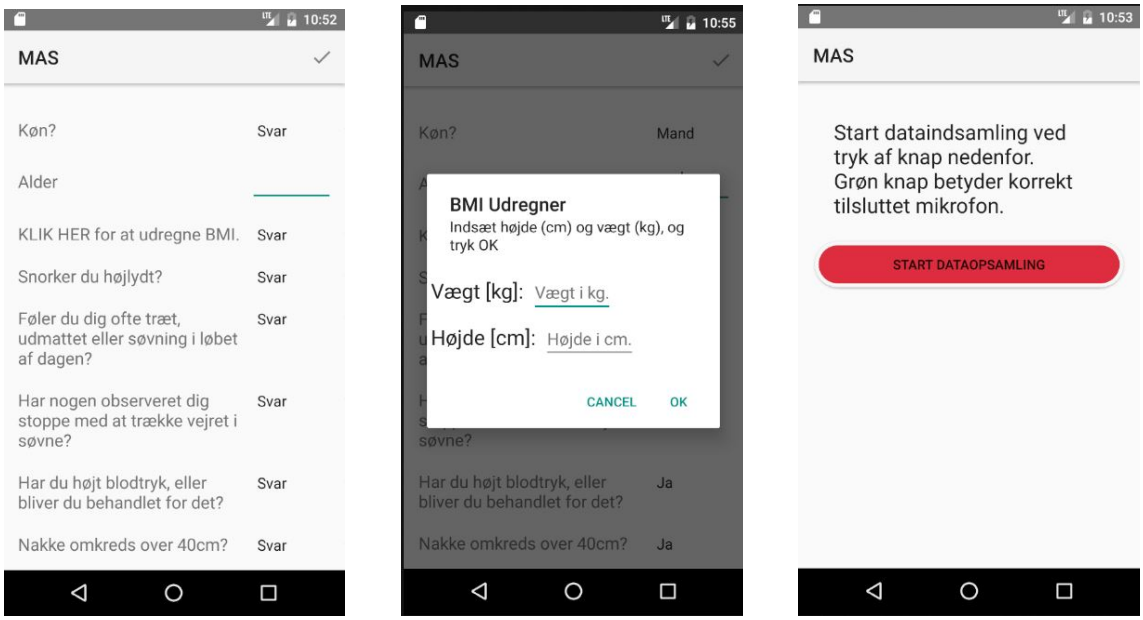


Fig. 5.3: Flowchart of the *"MAS"* application, developed to handle the data acquisition process. Blue boxes represent java(tm) activities, red boxes represents the different sensors, and the green represents data files (.txt and .pcm files). Arrows with floating text are followed as a result of user action, explained by the text, arrows without text floating text happens automatically. Lines represents continuous passing of information, such as the STOP-BANG answers being changed as the user answers the questions in Activity 1.

5.2.1 Graphical User Interface (GUI)

The main guidelines of the GUI were to make it as simple and easy to use as possible. Therefore, only functions relevant to the patients were implemented, making it impossible to change settings such as sample frequencies from within the application.

The complete GUI consisted of two activities and a dialog box, as seen in Fig. 5.4. The first activity is used to gather demographic information, regarding the patients via the STOP-BANG questionnaire. Furthermore, it sets the Patient ID to a unique name, based on the time and



(a) GUI of Activity 1, seen as soon as the application is opened

(b) BMI calculator, which shows when the user presses *"Calculate BMI"*

(c) Activity 2, entered when the tick in the upper right corner of Activity 1 is pressed

Fig. 5.4: The three GUI windows a user encounters when using the *"MAS application"*. Notice that it is in danish, but future versions will let the user choose between languages.

data, each time the activity is started or returned to. This is to 1) anonymize the patients and 2) making the application capable of performing multiple readings without closing the application. Activity 1 can be seen in Fig. 5.3. Activity 2 makes the patient capable of controlling the data acquisition time, using one button which works as an ON/OFF switch. Furthermore, the button also changes color depending on the microphone being connected to the phone or not (red stands for not connected and green stands for connected), see Fig. 5.4 for a graphical representation.

5.2.2 Accelerometer Sampling

The activity *AccServiceTimer.java* reads accelerometer measures continuously and saves them into three .txt files; xFile.txt, yFile.txt and zFile.txt. To extract readings from the accelerometer, the method *SensorManager* is used, which creates a pathway between the program and the sensors available on the phone. The sampling frequency can be modified by calling *SENSOR_DELAY_X*, where "X" is the chosen *MODE*. As explained in the state of the art chapter, the *MODES* available depends on the android phone and Table 5.1 lists the MODES in LG Nexus 5.

Mode	fs, screen ON	fs, screen OFF
Normal	15	4.96
Game	50	50
Fastest	198	198

Table 5.1: Accelerometer sampling frequency MODES of the LG Nexus 5, and the corresponding sampling frequencies. The *MODE* "Game" is chosen in the experimental design. Data from [30].

Since the source stating the sampling frequency of the LG Nexus 5 was a forum and not an official manufacturer, a series of tests were conducted to ensure the validity. The tests involved reading the time-stamps belonging to each measurement, and see if the interval between time-stamps was constant. With the sample frequency being 50 Hz, the time between time-stamps should be

$$t_{diff} = \frac{1}{fs} = 0.02s \tag{5.1}$$

This investigation confirmed that the sampling frequency was not completely constant, for some cases* the time difference were 00201 or 0.202 s berween two time stamps. In lengthy studies like this one, small deviations causes huge delays. One tests over 7 hours proved that the sampling frequency was 49.9206 samples pr. second. The delay compared to the chosen sampling frequency of 50.05 is:

$$fs_{diff} = fs_{actual}/fs_{MODE} \tag{5.2}$$

where fs_{actual} is the average sampling frequency that the accelerations have been sampled with, and fs_{MODE} is what should have been. This gives a relative difference of:

$$49.9206/50.05 = 0.9974 \tag{5.3}$$

The total delay this introduces can be calculated by multiplying this sampling frequency delay with the amount of seconds in one study, of approximately 7 hours:

$$t_{delay} = fs_{diff} * t_{inseconds} \tag{5.4}$$

Where t_{delay} is the total delay this change of sampling frequency introduces. Thus, the total delay can be calculated as:

$$(1 - 0.9974) * 60 * 60 * 7 = 65.3213s \tag{5.5}$$

This is proof that the sample frequency inconsistency must be corrected before using the data. If not, the accelerometer measurements will be skewed related to the audio readings, by more than one minute in recordings above 7 hours. The sampling frequency, showed no correlation with the screen being on or off..

5.2.3 Audio Sampling

The audio is recorded via the *AudioRecorderService.java* activity, which accesses the microphone and samples at a specified sampling frequency, using the *MediaRecorder* APIs. The samples are converted to bytes and written into a .pcm file named *audio.pcm*. The audio sampling frequency was set to 16 kHz.

5.2.4 Implementing Questionnaire

The implementation of the questionnaire is done in *Activity 1*, using *spinner variables* changeable via button clicks, and integer or string variable linked to *TextEdit* buttons. The graphical

representation can be seen in Fig. 5.4a, and the implementation of a BMI-calculator can be seen in Fig. 5.4b. The answers is saved in a .txt file named iFile, which contains general information of the data acquisition.

Question	**Answer**	**Output**
What is your gender?	[Male,Female]	[1,0]
How old are you?	Age	Years
What is your BMI?	BMI	m/s^2
Do you snore loudly?	[Yes,No]	[1,0]
Do you often feel tired, exhausted or sleepy during the day	[Yes,No]	[1,0]
Have anyone ever observed you stopped breathing while sleeping	[Yes,No]	[1,0]
Do you have high blood pressure or are you currently being treated for it	[Yes,No]	[1,0]
Does your neckcircumference exceed 40 cm? (shirt neck-size)	[Yes,No]	[1,0]

Table 5.2: The implemented questions of the STOP-BANG. Notice that question one and two have been modified, in order to gather more specific demographical information related to the subject. Furthermore, the questions implemented in the application are translated to Danish.

5.2.5 Application Output

The application output is a set of .txt files, containing, and the complete list can be seen below in Table 5.3.

File name	Approx file size 7 hours	Description
iFile.txt	1 KB	Information file
xFile.txt	13-16 MB	Accelerations in x axis
yFile.txt	13-16 MB	Accelerations in y axis
zFile.txt	13-16 MB	Accelerations in z axis
tFile.txt	22-26 MB	Time stamps in nanoseconds
audio.pcm	700-900 MB	Audio recordings

Table 5.3: List of files outputted by the MAS application and related information. The total amount of space of one recording was below 1 GB, and had less than 25 % of the battery capacity.

5.3 Experimental Procedure

A general procedure was created for the experimental trial, to be sure that it was conducted consistently and in a good practice. Each subject started by going through the standard procedure, associated with an CRM setup for diagnostic purposes followed by the opportunity of being part of the MAS experimental study [30]. The following steps describes the process of performing each individual data recording, for the subjects that were interested to take part in the project.

1. **Consultation**, with the researcher responsible of the scientific study (Bonnesen, M.P), involving a brief explanation of the project and presentation of the experimental equipment and relevant documents. These include the *Study Ethics document*, *Study information*, *written consent forms*[1] and *MAS user manual*. All documents can be seen in Appendix C.

2. **If a subject agrees**, to be part of the study, they sign the *written consent form* and receive the experimental equipment together with the rest of the documents explained in the previous step. Smartphone ID, name of subject and date from the *written consent form*, is notated before it is scanned to a file and filed in a room with a lock.

3. **Upon receiving the experimental equipment the next day**, the data is extracted and the phones are recharged. This is followed by a brief analysis of the acquired data, looking for signs of corrupt data; patient compliance regarding sensor placement, duration of data acquisition, questionnaire answers, and so on.

4. **Clean experimental equipment**, and gather the documents for the next subject. Wait for the assistants to conduct the analysis of the CRM, and then extract the results the sleep study; AHI and event notations.

All steps were written down in the Case Report File (CRF) (See Appendix C.3), to be sure that a good experimental procedure was followed.

[1] In Danish: Samtykkeerklæringer

5.4 Extracting CRM Analysis

Comparing the MAS performance to the *CRM* requires extraction of the results from the analysis namely the AHI. This is extracted by using the program *DOMINO - ver 2.7.0* made by *SOMNOmedics* especially for extracting information out of their device *SOMNOscreen*. The AHI was notated in the CRF, for each subject as soon as the recording had been analyzed.

In order to be able to investigate the usability of the signals acquired from the proposed method, all event annotations were extracted from *DOMINOS*. An algorithm gathered all event time-marks inside one matrix. The function made to do this is *EDF_event_extract.m* which outputs a comma separated .txt file with all events (see Appendix E for thorough CRM data extraction practice). Table 5.4 shows an example of the information related to two different events.

Event label	**Event Start (hour:min.:sec,ms)**	**Event Stop (hour,min.,sec,ms)**	**Length (s)**
Obstructive Apnea	05:10:35,325	05:10:55,700	20,3
Obstructive Apnea	05:11:12,982	05:11:40,000	28,0

Table 5.4: Example of event text file structure

To keep track of which CRM data belonged to which MAS reading, an excel document *Patient_info_Excel_v2* was created with information regarding each experiment; PatID, Smartphone ID, Name and start date. All data for each patient was put in a subfolder with their respective PatID inside a mother-folder called *DATA*. The CRM data was placed inside its own sub-folder, inside *PatID*, named *crm_data* (see data structure below).

5.4.1 MAS Analysis Tool

In machine learning there is a saying "garbage in, garbage out", meaning that poor data will lead to poor classification performance. This led to the creation of the script *CRM_MAS_events.m*, used for evaluating data segments of signals acquired on the MAS system, with the corresponding labels from the CRM analysis. By using the script *CRM_MAS_events.m*, the user can jump to notated *events* throughout the signal, and observe the corresponding audio segment, label vector, and feature vectors. The algorithm flowchart is seen in Fig. 5.5, and an image of the output can be seen in Appendix D.5. Furthermore, there is an option where *event-type* can be specified, so that the algorithm only shows segments with the specified event type.

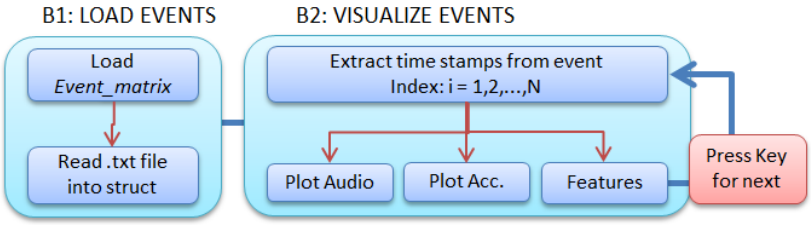


Fig. 5.5: Flowchart of how the *CRM_MAS_events.m* script works. The loop pauses after each iteration, and can be continued on button press. The *Event_matrix* was created using the script *CRM_data_extract.m*

6 Design of Advanced Obstructive Sleep Apnea Screening Algorithm

This chapter explains the methods used in the design of an advanced signal processing algorithm used for OSA screening. Each section will contain the methods used in each step of the algorithm; preprocessing, feature extraction, classification and validation. They will be presented with a brief introduction of the method, followed by where it is used and which changes have been made to the original method. Fig. 6.1, below provides a flowchart of the algorithm with each step of the algorithm.

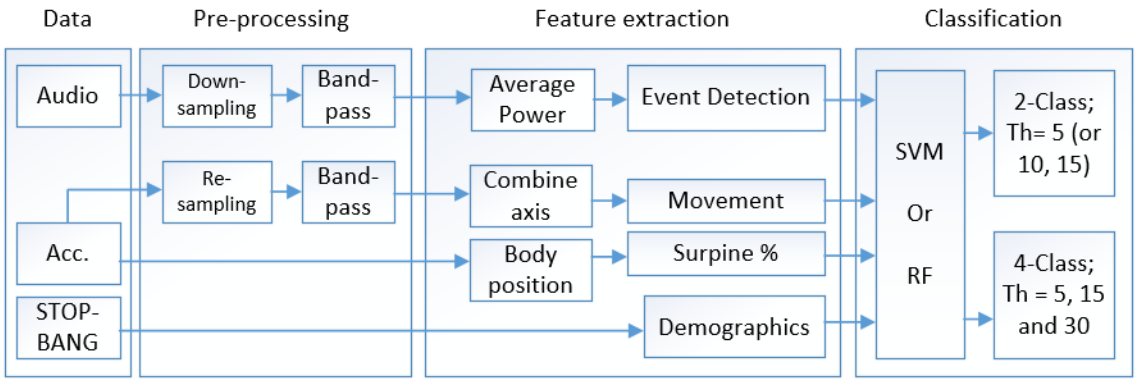


Fig. 6.1: Overview of the signal processing algorithm used for classifying subjects into categories. The specifications of each step, are explained in this chapter. *Th* is short for *Threshold*

6.1 Preprocessing

This section represents the preprocessing done on the two acquired signals; audio and accelerometer. Each section will contain which method was used, why it was chosen and it was implemented in the MATLAB algorithm. In order to analyze on the same time window as the CRM, a synchronization step is first used to synchronize the two signals and extract the time window to analyze on

6.1.1 Analysis Time Frame

The time frame on which the medical personnel do their data analysis is based on two marks; *LIGHTS OFF*, which is marked by the subject when he/she goes to bed and *LIGHTS ON* marked by the subject when he gets up in the morning. MAS has two marks describing the date and time of the recording; the *start* and *stop* times, both found inside the *iFile.txt*.

MATLAB Implementation

The time frame was extracted by first finding the lag between the *start* of the MAS recording and the *"LIGHTS ON"*-mark followed by a calculation of the length of the analysis segment in seconds, using the time length between *LIGHTS ON* and *LIGHTS OFF* (time and date). Both CRM marks are found in *markers.txt* and the MAS marks are found inside the *iFile.txt*. The image below in Fig. 6.2 illustrates how the MAS signal segment is extracted.

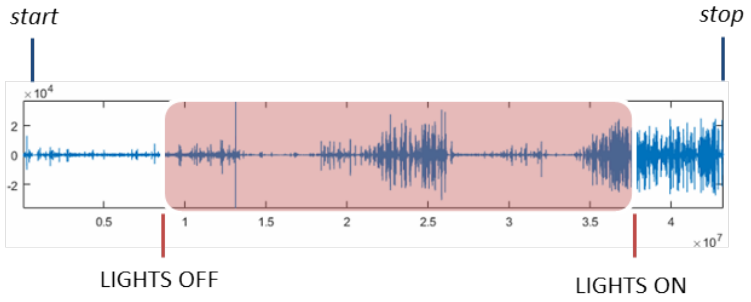


Fig. 6.2: Illustration of the process in which the relevant signal segment is extracted for analysis. *start* and *stop* is related to the MAS recording time, and *LIGHT ON/LIGHTS OFF*, the relevant CRM time frame. The red transparent square represents the MAS signal segment being extracted for analysis. The LAG is the time between *start* and *LIGHTS ON*.

6.1.2 Audio Signal

Audio signal preprocessing methods were based on detecting respiration like Al-mardini et al. [19] and Halevi M.J. [28], and the extracted frequency band were therefore chosen from 200 Hz to 1200 Hz. The noise removed this way are the well known *baseline wandering* and *powerline interferance*. In order to make the signal much less computational expensive to work with, it was downsampled from 16 kHz to 4 kHz.

MATLAB Implementation

The MATLAB function *decimate()* downsamples a signal by a specified *downsampling factor*, *r*. By default, the function uses a lowpass IIR Chebyshev filter of order 8, and a *normalized cutoff frequency* defined as 0.8/r (to avoid anti-aliasing). In order to downsample the audio signal to 4 kHz, a down-sampling factor $r = 4$ was used, resulting in a *cutoff frequency* of 0.2 Hz, and the signal being downsampled from 16 kHz to 4 kHz.

The first step of filtering the signal, was to create a *filter-object, d,* using MATLAB function *designfilt()*, which designs a digital filter from a set of input specifications. The specifications used can be seen below in Table 6.1.

Afterwards, the filtering was done using the MATLAB function *filtfilt()* with the input audio signal and filter-object *d*. *filtfilt()* performs a zero-face filtering in both forward and backward direction, thus resulting in *zero phase distortion*.

Filter type	Order	Fpass	Fstop
IIR Butterworth	10	200	1200

Table 6.1: Bandpass filter specifications for the bandpass filter used to extract frequencies related to respiration in the audio signal. The Phase and Magnitude response can be seen in Appendix D.2

6.1.3 Accelerometer Signal

The accelerometer signal was both used to estimate hearth rate (HR), respiration rate (RR), and used for extracting features. Before being filtered, the sampling frequency had to be corrected, which was done by resampling the signal into a constant sampling frequency of 25 Hz. The method used, is described in the MATLAB implementation.

Afterwards, signals was filtered into two different sets of X, Y and Z vectors. One set filtered to include frequencies related to respiration, determined to be from 0.1 Hz to 0.8 Hz, since this allows detection of a respiratory pattern between 6 and 48 breaths pr. minute. The other set was filtered to include heart rate related frequencies, determined to be from 0.8 Hz to 2.0 Hz, since this allowed HR between 48 and 120 to be detected. The higher *cutoff frequency* removes *Gaussian noise*, often seen in the higher frequencies of MEMS accelerometer measurements [35].

MATLAB Implementation

The resampling process startet by extracting the *time stamps* corresponding to each accelerometer value, from the *tFile.txt*. Since the stamps were in nanoseconds and notated according to how many nanoseconds have gone into the day, it had to be corrected to start from zero and be in the same units as the time *marks* of the CRM, namely seconds. This was done by using Eq. 6.1:

$$t_s = \frac{T - T(1)}{10^9} \tag{6.1}$$

, where T is the vector c*ontaining the time stamps in nanoseconds, T(1)* is the first time stamp and t_s is the new time stamp vector in seconds.

To perform the resampling of the signal, a MATLAB function called *resample()* was used. This function takes five inputs; input signal, time stamp vector, target fs_t, upsampling factor P and downsampling factor Q. The function uses a FIR lowpass filter with *cutoff frequency = 1/Q* before downsampling the signal, to follow the nyquist criteria. The input coefficients to *resample()* used in the proposed method was Acc_{signal}, t_s, $fs_{tar} = 25$, P = 40 and Q = 8.

The bandpass filtering was done using the same method as with the audio signal; *designfilt()* and *filtfilt()*. The filter specifications for both bandpass filters can be seen in Table 6.2.

Signal	**Frequencies related to:**	**Type**	**Order**	**Fpass**	**Fstop**
Accelerometer	Respiration	IIR Butterworth	40	0.10	0.80
Accelerometer	Heart Rate	IIR Butterworth	40	0.80	2.00

Table 6.2: Bandpass filter specifications for the bandpass filters used to extract frequencies related to respiration and heart rate. The Phase and Magnitude response can be seen in Appendix D.2

6.2 Feature Extraction

This section represents the feature extraction process of the classification algorithm. The purpose is to present the reader with the methods used, as well as the arguments for using the chosen features. Each subsection will contain an explanation of the feature choice, and the methods used to extract it from the signal using MATLAB. The feature *HR and RR estimation* was not used for classification, due to poor results (see Chapter 8 for further explanation), but the method will still be explained. The full list of features used for classificaion can be seen below in Table 6.3. The order of the features in the table follows the order they are presented.

Feature	**Data Source**	**Unit**	**Amount**
event_{m1}	Aud	$\frac{\text{events}}{\text{hour}}$	4
event_{m2}	Aud	$\frac{\text{events}}{\text{hour}}$	4
Apn_{mov}	Acc	Seconds	1
$\text{Surp}_{\%}$	Acc	%	1
BMI	STOP-BANG	$\frac{\text{kg}}{\text{m}_2}$	1
Age	STOP-BANG	Years	1
Severity	STOP-BANG	[1,2,...,6]	1

Table 6.3: List of features extracted from the signal in the feature extraction step of the algorithm. Since event_{m1} and event_{m2} provides four features each (due to multiple thresholds), leaving the total features to be 13.

The event detection algorithms used in the algorithm, event_{m1} and event_{m2}, was based on the work by Al-Mardini M. et al. [19] and Halevi M. et al. [28], and the similarities as well as the differences and implementation of the methods will be discussed in each subsection. The apn_{mov} feature was based on the work by Alwassim S. et al. [22] and the feature $\text{surp}_{\%}$ as well as the BMI and Age was based on discussions with project supervisor dr.med. Poul Jennum. The severity feature was implemented according to the study by Behar. J. [20].

6.2.1 Respiration Rate and Heart Rate estimation

Both the RR and HR are estimated by finding the peak frequency of a segment of M samples. When specifying the length of the segment, it is important to take into account the time and frequency resolution changes related to M values. A Large M, would increase the frequency resolution but decrease the time resolution, and vice versa. The peak frequency of a given segment is found by using fast fourie transforms (FFT) to transform it into frequency domain, from where the frequency with the highest peak is extracted.

MATLAB Implementation

First, the length of the segments, M, is defined. The segment length M, is 60 for RR estimation and 10 for HR estimation. The choices are based on the fact that the RR is in lower frequencies than the HR, and thus requires a higher frequency resolution. The frequencies of each segment is extracted using the MATLAB function *fft().m* to perform the FFT and *fftshift().m* to shift the zero-frequency spectrum to the center of the spectrum. The frequency component with the highest frequency was extracted using the MATLAB function *find().m*.

6.2.2 Event Detection

The two event detection features; Event_{m1} and Event_{m2}, uses the average signal power, *pAvg* to find events. The *pAvg* values can be calculated by taking M length window segments of the audio data, and extract the Power Spectral Density (PSD) estimation of the segment. The method used in this thesis is the *pwelch* method, which splits the segment into subsegments, finds the periodograms in each one, and average over them all [33]. The length of the window as well as the introduction of overlap are some ways of modifying the method.

This is done in one second windows throughout the signal, which results in a vector containing all the pAvg values from the one second windows, pAvg_{vec}. In the implemented method the window length in *pwelch* was chosen to be one second, in order to have a frequency resolution high enough to get the respiratory frequencies. The segment length was also chosen to be one second.

Event Detection, Method 1

The event detection algorithm event_{m1}, is inspired by the work of Al-Mardini M. et al. [19]. Changes have been made in order to make it more robust to noise, by using the signal *power* instead of the signal energy, to make low-amplitude respiration more distinctive from the ambient noise. Furthermore, instead of defining a threshold from the first few minutes of a recording, event_{m1} uses predefined thresholds to avoid the threshold being ruined by a noisy segment. The third change is the introduction of a upper threshold, used to label *snoring segments*. If a *pAvg* value is below lower threshold, it gets labeled as a *no-breath* value and given the index = 0. *pAvg* values between the two thresholds are labeled as *breath* windows with index = 1 and *pAvg* values above the upper thresholds are labeled *snore* with index = 2.

Afterwards, the label vectors are investigated for sequences consisting of 10 or more consecutive *no-breath* windows (windows labeled as *0*). For each time occurrence, the whole segment of consecutive 0's are labeled as *event* using index = -1, and one is added to an event counter, event_{c}. The last step is to find the rate of event in hours, using Eq. 6.2:

$$\text{event}_{\text{m1}} = \frac{\text{event}_{\text{c}}}{\text{length}_{\text{sig}} * \frac{1}{\text{fs}_{\text{aud}} * 3600}} \tag{6.2}$$

, where $\text{length}_{\text{sig}}$ is the length of the signal in samples, fs_{aud} is the audio sampling frequency and event_{m1} is the average amount of events pr. hour.

Event Detection, Method 2

The second audio event detection algorithm, event_{m2}, is based on the information that an event is often followed by a big increase in airflow due to the body needing to compensate for the lost

oxygen during the event. This will be reflected in the audio signal by a big increase in the signal energy. event_{m2} is about detecting this change of signal energy, and is a modification of the feature *SuperSnore* used by Halevi M. et al. [28]. Changes have been made in order to take into account different noise sources. One is when the upper airways transits between being almost closed to being closed, resulting in high pitch high energy "gasps" sounds. To take this into account, the *skewness* will be the average value from all samples inside a window of i samples forward divided by j samples backwards in the signal. The large difference between different events that *pAvg* provides, was an advantage in the previous event finding algorithm, but not when used in this method. The reason is that these "gasps" can have *pAvg* more than a 10-fold as large as snores, resulting in a lot of false events. To make the method less sensitive to this noise, the *natural logarithmic* values of *pAvg* values is used instead. The parameter *Skew* is calculated as in Eq. 6.3:

$$\text{Skew}_{\text{vec}}(\text{s}) = \frac{\sum_{\text{i}=1}^{\text{N}} \log(\text{pAvg}_{\text{vec,s+i}})}{\sum_{\text{j}=1}^{\text{M}} \log(\text{pAvg}_{\text{vec,s-j}})} \tag{6.3}$$

, where s is the vector index in which the *Skew* values is calculated, i is number of *pAvg* index values before s, and j is number of *pAvg* after s. The result is a vector Skew_{vec} with Skew values for each index of pAvg_{vec}, except the values from start to the j index. See Fig. 6.3, for an illustration of an example of how the *Skew* values reflects an audio signal.

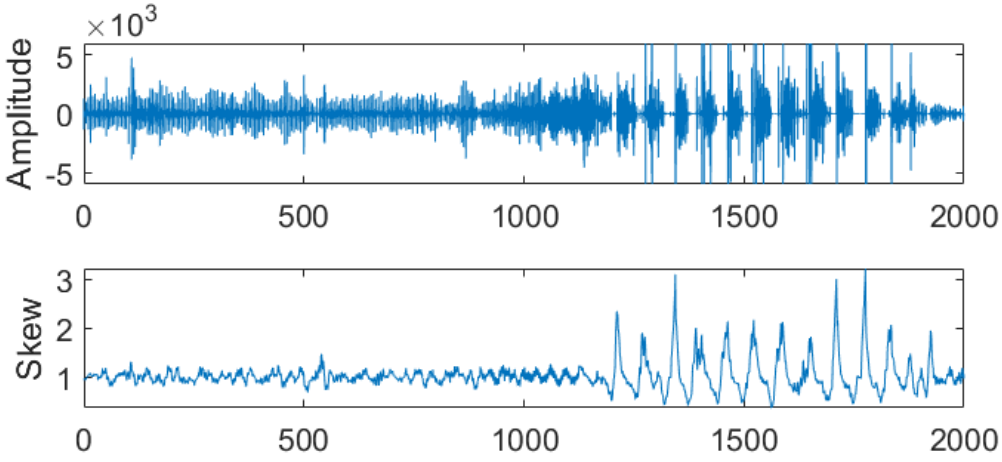


Fig. 6.3: Illustration of an audio signal (upper plot) and corresponding *Skew*-values (lower plot). Events are reflected as big increases in *Skew*-values. A lot of events are occurring from 1300 sec. to 2000 sec. mark.

Since the increased breathing step after an event can have multiple local maxima, a peak-finding algorithm must be used to extract peaks with a specified minimum distance to each other. Finally, the number of peaks are counted as event_{c}, and Eq. 6.2 is used to find the rate of events, which is the feature event_{m2}.

MATLAB Implementation

The structure of the MATLAB implementation of the both the event-detection algorithms are shown below in Fig. 6.4.

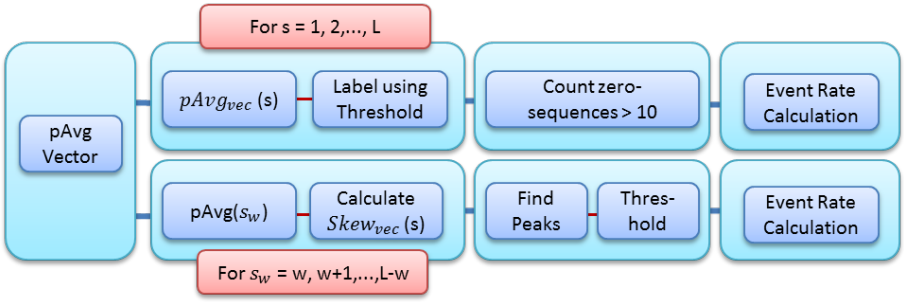


Fig. 6.4: The process of each event-finding algorithm. The top row is event_{m1} and the bottom row is event_{m2}. L is the length of pAvg_{vec} vector, w is a defined window.

The MATLAB function pwelch() calculated the PSD using the *pwelchs* method. The window length, w, was equal to the sample frequency of the audio signal, w = 4000 with no overlap. To calculate spectral parameters a MATLAB function was generated, called *spec_feat()*. It takes input PSD segments and calculates the pAvg values of the segment. The result is a vector, pAvg_{vec} with pAvg values for each second of the signal.

The event_{m1} events were found by looping through the pAvg_{vec} vector and label each value using the defined thresholds, into a new vector $\text{labels}_{\text{m1}}$, while counting number of events. event_{m2} events were found by looping through the signal, extract windows of data points based in the second index s and calculate the skewness values using Eq. 6.3. The result were a vector containing *Skew* values named Skew_{vec}. The MATLAB function *findpeaks().m* was used to extract peaks from the Skew_{vec} with a minimum distance between peaks as 10 seconds.

Both measures used an implementation of Eq. 6.2 to calculate the event rate index. A total of four thresholds were used for each event detection algorithm, see Table 6.4, resulting in a total of 8 features.

	Threshold 1	**Threshold 2**	**Threshold 3**	**Threshold 4**
event_{m1}	5000	$q_{60\%}$	$q_{50\%}$	$q_{40\%}$
event_{m2}	$q_{99\%}$	1.8	2	2.5

Table 6.4: Table listing the different thresholds used in the two event detection algorithms. Each threshold is used individually to calculate a feature value for the event.

The thresholds were defined on the basis of analyzing the signals using the *Signal Visualization tool*. The reason that the threshold from Al-Mardini M. et al [19] was not used, were because it was found not be to robust enough.

6.2.3 Apneic Movement

To reflect the intensity of the body movement, the method *Signal Vector Magnitude* was used. This combines the movement in all axis into one value, reflecting the total movement, and is calculated as

$$C_{val}(i) = \sqrt{x(i)^2 + y(i)^2 + z(i)^2} \tag{6.4}$$

, where x,y and z are accelerometer measurements for each axis, at a given sample i and C_{val} is the corresponding Signal Vector Magnitude value. When the value of c_i exceeds a specific threshold, a segment is defined as "Apneic Movement".

MATLAB implementation

The signal vector magnitude was calculated by inserting the accelerometer vectors; x, y and z, into Eq. 6.4 to get a vector C_{vec}, containing C_{val}-values for each sample. Since the feature *apneic mov* is in seconds, the vector C_{vec} is looped through in window segments of 1 second (window length $w = 25$ *samples* due to the $fs_{acc} = 25$), calculating the average C_{val} value in each segment, and thus creating a new vector $C_{val,s}$. Then values above a threshold are extracted using the MATLAB *find().m* function, and *Apneic movement* is defined as the number of values in $C_{val,s}$ above a threshold defined as the 95 % quantile of the values inside $C_{val,s}$.

6.2.4 Body Position

The theory used to extract the body position from a set of 3D accelerometer readings is based on the knowledge of accelerometer placement relative to the body. This is due to the fact that the maximum gravitational influence on a accelerometer axis is $\approx 9.8\frac{m}{s^2}$. This is large compared to the forces related to movement while sleeping, which is generally below $2\frac{m}{s^2}$ even when turning around and only $0.1 - 1\frac{m}{s^2}$ while breathing. A key piece of information that is needed is the position of the accelerometer in space. The smartphone, containing the 3D accelerometer, was placed on the *suprasternum* with the screen turning outwards and jackstick upwards. See Fig. 6.5 for an illustration of the gravitational influence on the accelerometer measurements.

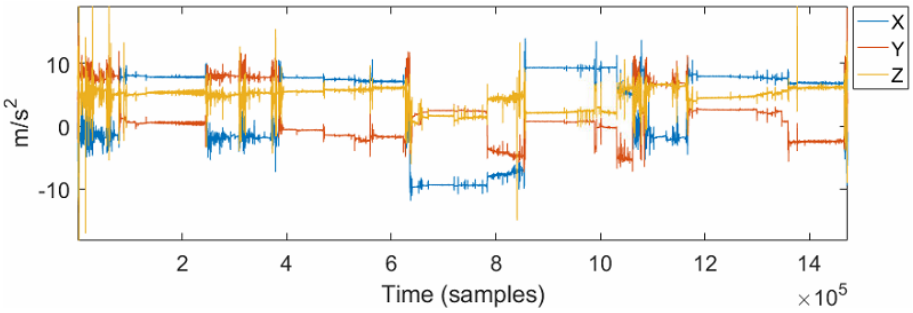


Fig. 6.5: Graph of the raw accelerometer measurements on all three axis; X, Y and Z, during a full nights sleep. The data is from *Pat 4*.

Notice, that in Fig. 6.5, the big changes in signal amplitude is from the *gravitational component* when the sleep position is changed, where the smaller is due to respiration and body movement not related to sleep position change. Since it is possible to sleep in for example *surpine* position but with a small tip towards the *right lateral*, the *gravitaitonal component* is not always $\approx 9.8\frac{m}{s^2}$.

It is important to note that the feature extraction of the body position must be done on the raw signal, since a highpass filter would filter the gravitational influence out of the signals.

MATLAB Implementation

The implementation in MATLAB was done by creating a function bodyp().m. The function looped through the accelerometer vectors; X, Y and Z, in windows of length N = 4, and calculated the average value in each axis; $x_{avg}(n)$, $y_{avg}(n)$ and $z_{avg}(n)$. After each calculation, the sleep position at the current time was determined on the bases of a set of rules, implemented as *"if-statements"*. An overview of each rule and the corresponding body position can be found below in Table 6.5.

Body position	**Rule 1**	**Rule 2**	**Index**
Surpine	Z > X and Y	Z > 0	1
Prone	Z > X and Y	Z < 0	2
Left Lateral	X > Y and Z	X < 0	3
Right Lateral	X > Y and Z	X > 0	4

Table 6.5: List of rules which the MATLAB code is based upon and the corresponding indexes related to each sleep position. X, Y and Z are acceleration magnitude values.

The feature used in the MAS algorithm, is the percentage of the whole night's sleep spend in surpine sleep position and is calculated using Eq 6.5:

$$\text{Surp}_{\%} = \frac{\text{Time}_{\text{surpine}}}{\text{N}} \tag{6.5}$$

, where $\text{Time}_{\text{Surpine}}$ is the time spent (samples) in surpine sleep position and N is the signal length in samples.

6.2.5 Questionaire

The features extracted from the questionnaire were *BMI*, *age* and *severity*. Age, hight, weight and STOP-BANG answers were inputs in the smartphone application. Age is therefore directly extracted from the *iFile.txt*, BMI is calculated by extracting hight and weight and inserting it in Eq. 6.6:

$$\text{BMI} = \frac{\text{w}_{\text{kg}}}{(\text{h}_{\text{m}})^2} \tag{6.6}$$

, where w_{kg} is weight in kg and h_{m} is height in meters. The *severity* is the number of questions in the STOP-BANG answered as "yes".

MATLAB Implementation

Age, hight, weight and STOP-BANG answers was read from the *iFile.txt* using MATLAB function *textscan().m*. BMI was calculated using Eq. 6.6. Since BMI and *age* had to be binary in order to be included in the *severity* feature, thresholds used in the STOP-BANG was implemented; $\text{age}_{\text{threshold}} = 50$ and $\text{bmi}_{\text{threshold}} = 35$.

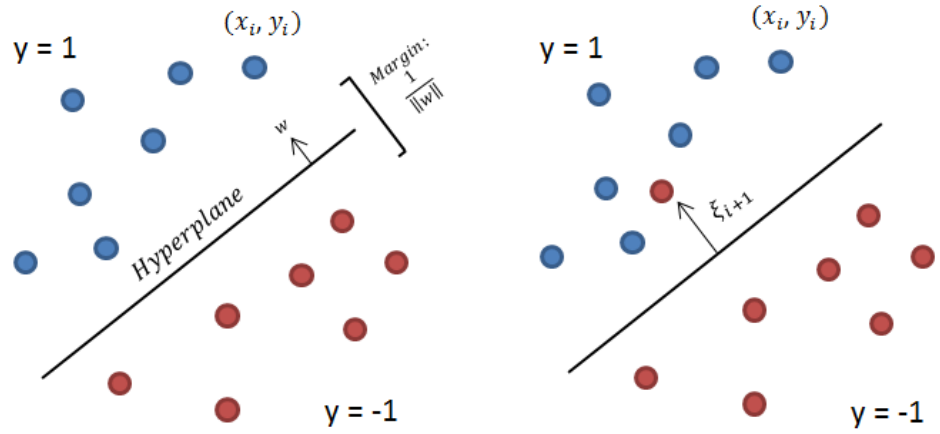
6.3 Classification

This section contains theory related to the two classifiers used for classification: Support Vector Machine (SVM) and *Bagged Decision Tree*. Both Behar J. et al. [20] and Shi C. et al. [25] used

SVM's to classify their data. The reason that the study also tests the use of Bagged Decision Trees (BDT) is due to the fact that SVM's tend to have reduced performance when working with multiclass problems. Since they are binary classifiers, the only way to expand them for multiclass-problems are by creating multiple SVM's. This is not the case with the BDT classifier, which is easily expanded to multi-class problems simply by adding a class in the training set. Another difference between the two classification methods is that features must be normalized or standardized before being used in the SVM [39][38], opposite to the BDT where it is not needed.

6.3.1 Support Vector Machine

The support vector machine is a classification method that tries to find a hyperplane that separates classes in a M-dimensional features space. Lets first look at an illustration of the problem for M = 2 classes, classes y = -1,1.



(a) Illustration of basic concepts of *margin* and *hyperplane* using a perfect separable dataset.

(b) Illustration of basic concepts of *emperical risk* using a dataset not perfectly separated.

Fig. 6.6: Illustration of a Feature space with M = 2 features and N = 14 data-points each belonging to one of two features C_1 or C_2. The *blue* and *red* dots are two different classes, and the line between them is the *hyperplane*.

To predict a new data-point x, into one of two classes y = –1, 1, Eq. 6.7 is used:

$$f(x, w, b) = \text{sgn}(w \cdot x + b) \tag{6.7}$$

This results in the data point being classified into class 1 or -1. The aim of the training step is thus to create a *hyperplane* that separates the classes well. Intuitively, the best possible separation must be when the margin $1/\|w\|$, is as large as possible, which can be re-written to a minimization of the term $\frac{1}{2} * \|w\|^2$, called the *complexity*. However, classes are rarely as well-separated as in Fig. 6.6(a), and sometimes a degree of error must be accepted (see Fig. 6.6(b)). The error term ξ_i is the perpendicular distance from the data-point i to the hyperplane, and the total error is called the *emperical risk* [39]. The trade-off between the *simplicity* and the *emperical risk* is the core of the SVM training step. This can be seen as the optimization problem in Eq. 6.8, known as the canonical representation:

$$y_i * (w \cdot x_i + b) \geq 1 - \xi_i \quad i = 1, ..., N \tag{6.8}$$

, where ξ_i is the is the *risk* at data point *i* and y_i is the class correct data-point class label. Data points for which the equality holds, are *active* (called the *support vectors*) and if not, they are *inactive*.

In order to solve the problem of both minimizing the *complexity* and the *emperical risk*, we introduce the so called *lagrange multipliers* $\alpha \geq 0$ for each data point i. They are useful because they make it possible to weight the relevance of each *i* data-point [39]. An outlier in the data-set would not influence the performance, since the ξ belonging to it would be near zero. The next step is to introduce the *Lagranian function* seen in Eq. 6.9:

$$min \quad L(w, b, a) = \frac{1}{2} \|w\|^2 + C * \sum_{i=1}^{N} \alpha_i (y_i * (w_i \cdot x_i + b) - 1 + \xi_i) \tag{6.9}$$

, where *C* is a trade-off parameter introduced in the optimization problem between the *complexity* and the *emperical risk* and $0 \leq \alpha_i \leq C$. Notice that the α can control the relevance of the constrains introduced by a data-point *i*, as mentioned previously. The next step is to use Eq. 6.9 to build the so called *duel representation* seen below in Eq 6.10.

$$L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j * x_i \cdot x_j \tag{6.10}$$

Eq. 6.10 shows that the optimization only depends on the dot product of pairs of samples and the lower the trade off parameter *C* is defined, the less important errors will become. Since the optimization only depends on the dot product, it can be transformed into linear space using a *kernel function*. This fact, is one of the advantages of the SVM. The linear kernel can be seen below in Eq. 6.11, and an image of the transformation can be seen in Fig. 6.7.

$$K(x_i, x_j) = \Phi_i \cdot \Phi_j \tag{6.11}$$

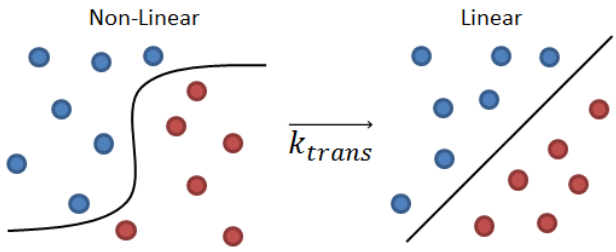


Fig. 6.7: Illustration of a how a non-linear classification problem becomes a linear problem by using a kernel function

Multiclass SVM

Since the SVM is fundamentally a binary classifier, the only way of using it in a multi-class problem, is to create multiple SVM's [38]. There are two different approaches to this; the *one-vs-one* classification and the *one-vs-rest* classification. The *one-vs-rest* constructs M separate SVMs, where M is number of classes. Each SVM is trained to distinguish one class from the rest, by merging all but one class into one *mother-class* and then do the binary classification between each class and the *mother-class*. This is done for each class, and the decision is made based on which class the data point is most likely to be in, when its compared to the rest. The *one-vs-one* simply creates K classifiers, and makes the classification pairwise with each class, assigned the datapoint to the class with the highest probability. Both methods are illustrated in Fig. 6.8 [38].

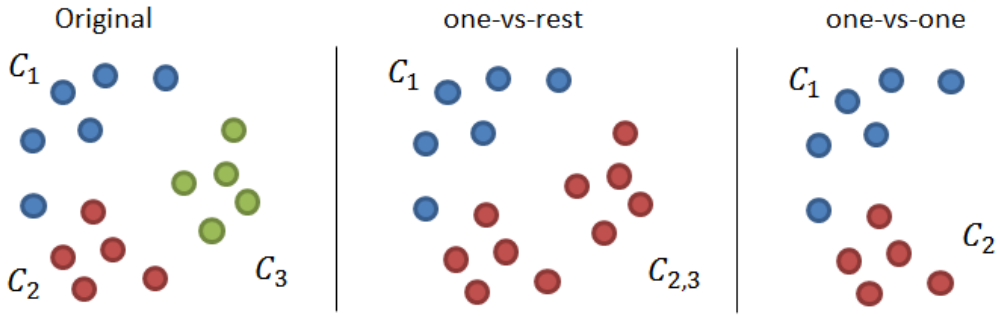


Fig. 6.8: Illustration of the *one-vs-rest* and *one-vs-one* multi-class SVM approaches.

MATLAB Implementation

The MATLAB build in function *fitsvm()* is used to train the SVM, using an input feature vector Feat_{vec}, with the corresponding labels Lab_{vec}. The kernel is set to be a *linear kernal*, feature standardization is set to be *ON* and the box constraint *boxC = 1*. The *boxC = 1* is the trade-off parameter, *C*, introduced earlier in Eq. 6.9. Increasing *boxC* makes for a stricter class separation since it increases the weight of miss classifications.

In order to use the SVM for the *multi-class* classification problem, a MATLAB build-in function called *templateSVM()*, is used instead of *fitsvm()*. It has the same inputs as *fitsvm()* but an output structure that fits the input to another MATLAB build in function *fitcecoc()*. *fitcecoc()* is used to fit multiclass learners. In the case of the MAS algorithm the type of learner is defined as SVM, and the coding is *one-vs-one*. The function, *MAS_SVM_class()* was created in MATLAB for training and testing binary and multiclass SVM models based on the above methods.

6.3.2 Bagged Decision Trees

Bagged Decision Trees (BDT) are widely used classifiers, since they can easily be applied to a wide range of classification problems, even with small sample sizes. It is an extension of the *Decision Tree* classification method, in that it "grows" multiple decision trees from data sub-sets, and then aggregates the decisions of all of them to make predictions [36]. Therefore to understand the BDT, one must start by understanding the theory behind the *Decision Tree*.

Decision Tree

The decision tree is a simple classifier that works by splitting the data into groups until a split assigns all data to the correct class or a stop-criterion is met. It is called a tree due to the

tree-like structure it has, illustrated in a simple decision tree classification seen below in Fig. 6.9 [37].

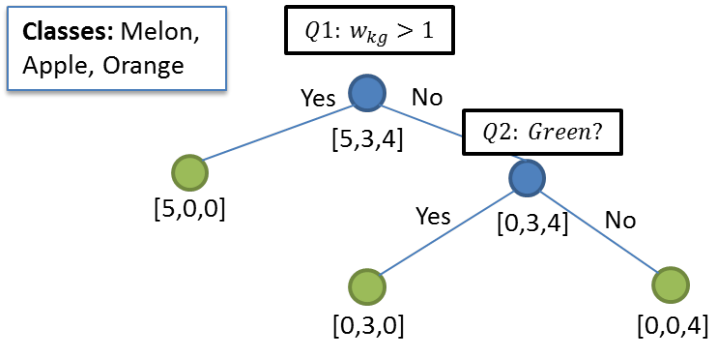


Fig. 6.9: Illustration of a simple 3-class decision tree classification, defined in the blue box. The blue dots are **nodes**, green dots are *Leaves*, text inside the black boxes are *split criterion* and vectors beneath nodes and leaves are class distributions

This is a simple example of a decision tree classification problem, classifying fruits into one of three classes; melon, apple or orange using two features. The first feature (continuous), w_{kg} distinguishes melons from apples and oranges perfectly, therefore resulting in a *leaf* (green dot). The second feature, distinguish apples and oranges perfectly, thus ending the classification. The idea is to keep *growing the tree* (adding nodes) until all branches are *leaves* or until a specified stop criterion (tree depth) is reached.

Splitting Criterion

In order to do the best split of each node, the term *purity* is introduced, which reflects how well the data are separated in a split. Multiple *purity measures* exists, below in Eq. 6.12 are shown how to calculate the one used in the algorithm, namely the **Gini-index**:

$$\mathrm{Gini(t)} = 1 - \sum_{\mathrm{i=0}}^{\mathrm{c-1}} [(\mathrm{p(i|t)}]^2 \tag{6.12}$$

, where p(i|t) denotes the fraction belonging to class i, at a given node t [37]. A node with perfect class separation would cause the summation term to become one, and thus making the *Gini-index* zero, meaning perfect *purity* in the split. There are multiple ways of defining the threshold for the split, but one is to test multiple bins of dataset, for example 25 %, 50 % and 75 % quantile and evaluate the performance of each bin, using *purity* measure.

Decision Tree Bagging

Deep decision trees tend to over fit the data, due to a very low bias and high variance. The *bagged decision tree* is a way of reducing the variance, by making decisions based on multiple decision trees. The standard BDT creates N decision trees using creating evaluating all features for finding the best splits, for each tree. The share of the decision trees arriving at each class is calculated, and the class with the highest share of decision trees is chosen as the prediction. Below in Fig. , three decisions tree from a bag of decision trees.

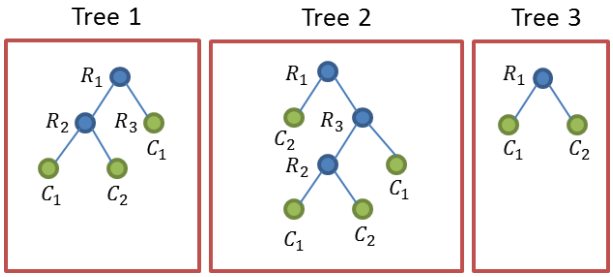


Fig. 6.10: Illustration of three decision trees from a bag of decision trees, in a simple 2-class classification problem. C_1 and C_2 is the class separation a *leaf node* predicts the input data to be in. R_1, R_2 and R_3 is the rules that separates data in each node.

Notice that trees from the same bag do not necessarily have to be the same size, since they stop growing as soon as all classes are separated perfectly, or the stopping criterion is met. If *Tree 1* and *Tree 2* both classified a data-point into C_1 and *Tree 3* classified it into C_2, the scores would be [C_1,C_2] = [0.66,0.33]. Thus the data would be classified as C_1.

MATLAB Implementation

The *bagged decision tree* algorithm is implemented in MATLAB by using the build-in MATLAB function *fitensamble*. This function allows the programmer to specify the type of classifier, how many of them there should be put in the "bag of classifiers", and the type of problem there is at hand (classification or regression). The classifier was put as *Decision Tree*, with *number of learners* being 50 using the default *Gini* split criterion. The function, *MAS_BDT_class()* was created in MATLAB for training and testing BDT models based on the above methods.

6.4 Validation

To validate the classification processes, N-fold cross-validation is used. If N = 5, the algorithms would do steps, where it trained the classifier using 4/5 of the data, and then tested it on 1/5 of the data. Due to the limited dataset of 23 subjects, *leave-one-out* crossvalidation is used, and therefore N = 23. If the dataset was bigger, it could also be interesting to extract part of the dataset, only to be used as test data. It is always a good idea to test a method using completely unknown data, if the dataset is big enough. The *confusion matrix* method is used for evaluating the performance of the classifier. It compares values predicted by the classifier with the actual labels and thus provides an overview of the overall classification, see Fig. 6.11.

The size of the confusion matrix is MxM where M is the number of classes in the classification problem. From Fig. 6.11, one can conclude that a perfect classification only have positive values in the diagonal. To analyze the results on in the *confusion matrix*, two performance measures were used; *Sensitivity* and *Accuracy*. *Sensitivity*, also called True-Positive-Rate is important because it measures the amount of sick people being diagnosed as sick. The second measurement, *accuracy*, reflects the overall performance of the classifier by comparing the number of true predictions with the number of points in the dataset [37]. The two measurements are calculated as in Eq. 6.13 and Eq. 6.14, using variables from the confusion matrix.

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6.13}$$

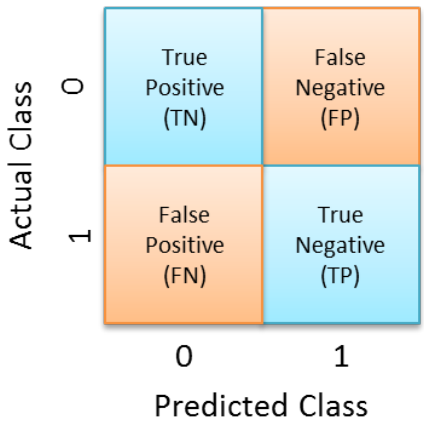


Fig. 6.11: Illustration of *Confusion Matrix* structure for a binary classification problem, with text describing each field inside the matrix.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TN} + \text{FP} + \text{FN} + \text{TP}} \tag{6.14}$$

MATLAB Implementation

The validation step was implemented by creating the script classification_script.m. The script includes both of the classification functions, *MAS_SVM_class()* and *MAS_BDT_class()*, and the user can specify model parameters, as well as the type of crossvalidation used. It can also be used to specify which features should be used for classification.

7 Results

This chapter contains the results obtained using the MAS system and is split into three sections. The first section describes the database; number of subjects, demographic information, file size etc, and demonstrates how *respiratory events* related to OSA is seen in the acquired signals. The second section visualizes both event-detection algorithms and some feature characteristics related to the labels. The third and last section, evaluates the classification performance of all the classification problems, with both classifiers.

7.1 Acquired Data

This section describes the acquired dataset, and then visualizes some *apneic* and *hypapneic* events using all of the acquired signals; audio, X-, Y-, and Z-accelerations.

7.1.1 Dataset

The dataset includes 23 subjects; 17 men and 5 women. 7 Subjects did not perform the data acquisition, of which 4 changed their mind related to being part of the study (Pat 11, 17, 20 and 21), one forgot to insert the microphone (Pat 7), and 2 forgot to turn on sampling when going tog bed (Pat 27, Pat 28). Table 7.1 below, shows the statistics of the subjects that did conduct the data acquisition.

	Mean	**Std**	**Min**	**Max**
BMI	29.4	5.3	20.4	41.4
Age	57.4	14.3	22.0	75.0
AHI	26.2	21.4	2.6	82.6

Table 7.1: Table with statistics related to the subjects in the dataset.

Using the AHI-labels from Table 2.1, 5 subjects were healthy, 4 had *mild* OSA, 5 had *moderate* OSA and 9 had *severe* OSA. The analysis was scored by medical professionals at *Rigshospitalet Glostrup*.

7.1.2 Signal visualization

This section visualizes the signals acquired through the MAS signal acquisition method. The purpose is to present the reader with different findings, found by using the before mentioned *MAS Analysis Tool* in section 5.4.1.

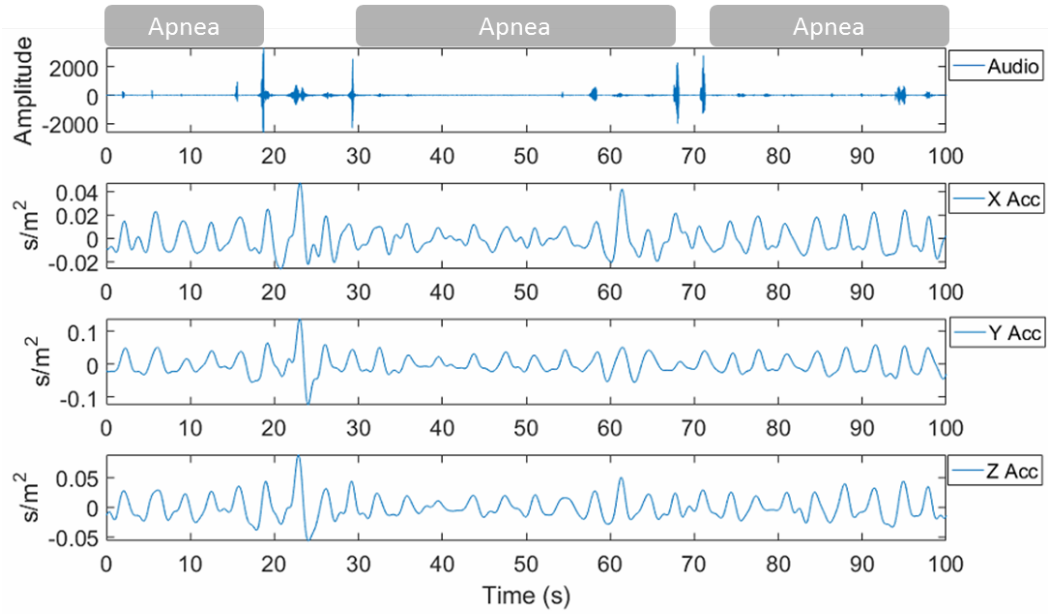


Fig. 7.1: Plot of an audio segment with labeled *apneic* events, reflected as signal segments below the gray boxes. From subject *Pat 13*, while sleeping in *left lateral* sleeping position.

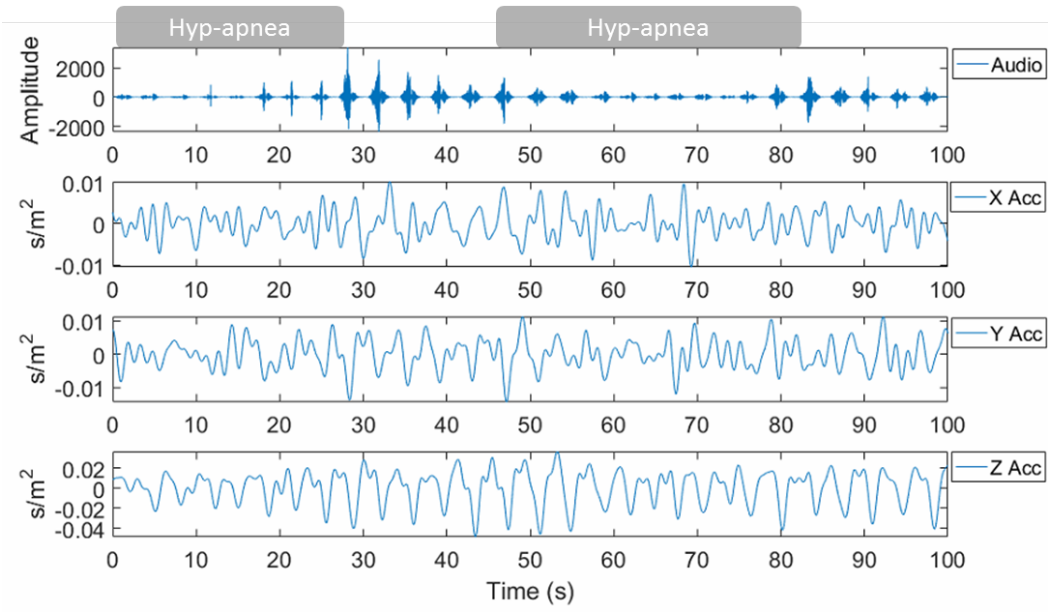


Fig. 7.2: Plot of an audio segment with labeled *hypapneic* events, reflected as signal segments below the gray boxes. From subject *Pat 13*, while sleeping in *left lateral* sleeping position..

From the audio signal in Fig 7.1 and Fig. 7.2 it is possible to see the different characteristics of the two types of events. The audio segments containing *apneic* events in Fig. 7.1 have very low signal amplitudes during the events. The only visible peaks are the before-mentioned "gasps", seen at 18 sec., 58 sec and 96 sec. marks. This is opposite to the audio signal containing *hypapneic* events, where a respiration pattern is clearly visible, but with changing signal amplitudes. Another

interesting finding can be seen the accelerometer readings of the two figures. The respiration pattern is very noisy in the X- and Y-axis of Fig. 7.2, while it is clear in all three axis in Fig. 7.2. Since both segments are from the same subject, this indicates that the accelerometer does move location during the night, sometimes to locations with reduces quality of the acquired signal quality.

7.2 Feature Extraction

This section is meant as a way of presenting the reader with interesting findings related to the feature extraction process. The first subsection presents some of the results that lead to the decision of not including the RR and HR estimation in the final feature set. Afterwards, the results of the two event-finding algorithms event_{m1} and event_{m2} will be visualized and different problems encountered in each method, will be stated. Finally, a correlation plot between the features and the AHI labels will be shown and commented.

7.2.1 RR and HR Estimation

The following two figures, Fig. 7.3 and Fig. 7.4, represents some general trends seen in the HR and RR estimation, using data acquired from Pat 10. They are calculated using different window values, since time and frequency resolution necessary for a good estimation is different between the two.

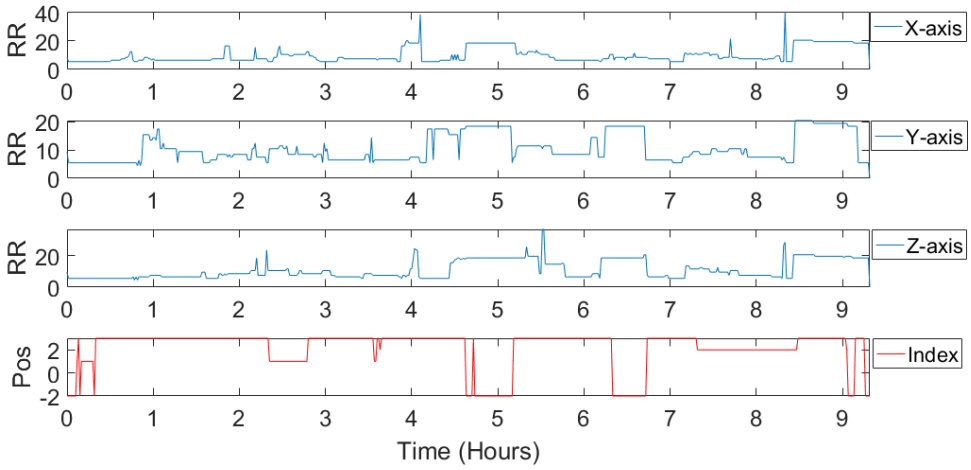


Fig. 7.3: Results of estimating the respiration rate using the accelerometer data acquired from Pat 10. The time resolution is 60 seconds steps between each RR estimation.

The RR estimation signal had large segments of more or less constant respiration rate, as seen in Fig. 7.3. Some large segments has respiration rates close to the minimum detectable of 6 breaths/min, a trend seen in multiple readings, indicating poor fixation of the smartphone. Another observation is that the different axis, x, y and z estimates different RR rates. Fig. 7.4 shows the result of heart rate estimation on data from the same patient as in Fig. 7.3.

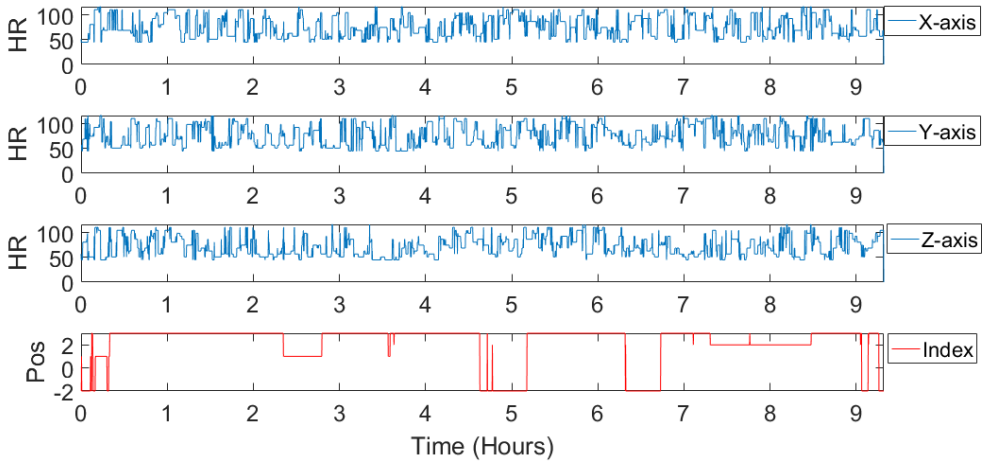


Fig. 7.4: Results of estimating the hearth rate using the accelerometer data acquired from Pat 10. The time resolution is 10 seconds steps between each HR estimation.

Fig. 7.4 shows the general trend of very noisy estimations of heart rate using all three axis; x, y and z. Generally it was determined that further analysis of the HR and RR estimates had to be done before anything can be concluded.

7.2.2 Event Finding

This subsection illustrates findings related to both event-detection algorithms, by visualizing specific events and commenting on the strength and weaknesses of each algorithm. This subsection illustrates findings related to both event-detection algorithms, by visualizing specific events and comment on the strength and weaknesses of each algorithm.

Event Method 1

Since the snore segments is not relevant for the event-detection they are all labeled as *breath* (index = 1), in order to make the plots easier to interpret. Furthermore, for visualization purpose, the labeling is amplified by 4000, meaning that each event label (*breath*, *no-breath),event* is equal to [4000, 0, -4000] instead of [-1,0,1]. This way, it is possible to see the labeling on the same plot as the audio.

The two upper plots in Fig. 7.5 illustrates how the trend in *pAvg* values change under *apneic events*. Notice how stable its changes between *breath* and *no-breath* windows in the first 100 seconds, reflecting normal respiration. From the time interval 200 sec. to 850 sec, a number of *apneic events* occur, reflected by big dips in the *pAvg* values, (seen in plot number 2 from top). Plot 3 and 4 (seen in plot from top) are subsegments of the above plots shown in greater detail. Two *apneic events* are correctly labeled; one from 690 - 710 sec. and one from 747 sec. to 778 sec. The one from 618 sec. to 651 sec. is not labeled correctly, which introduces a central problem in the Event_{m1} method, namely noise segments in the middle of events.

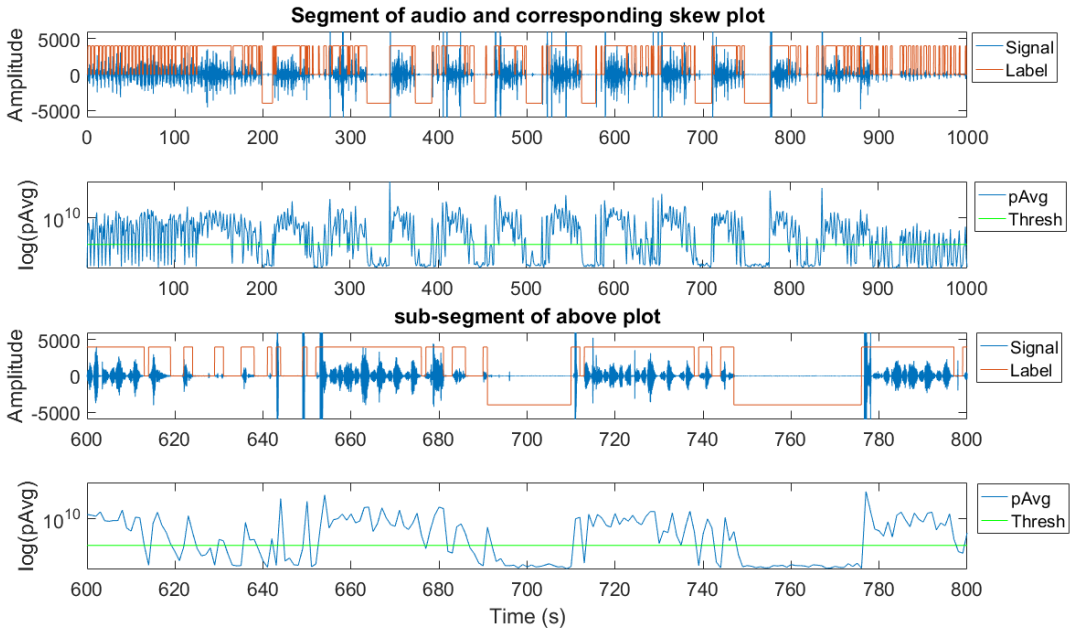


Fig. 7.5: Illustration of how events are labeled using Event_{m1} method, and data from patient 32. The 1st plot (from top) is a labeled (threshold = $Q_{0.60}$) signal segment of 1000 seconds length, and the 2nd plot (from top) is the corresponding *pAvg* values. The 3rd and 4th plots (from top) are segments of the above two plots, shown in great detail to provide a more detailed look into the event labeling.

Event Method 2

One of the reasons the second event-finding algorithms was developed, event_{m2}, was to cope with noise in the signal, and thus be able to correctly label the events. Below in Fig. 7.6 is an illustration of how the same segment from Pat 32 is labeled using Event_{m2}.

Notice that the mechanisms of labeling events in the Event_{m2} event-finding method is different from Event_{m1}, in that it only labels locations where it finds an event, instead of labeling a full *event* segment. Also notice that during the first 100 seconds, the *Skew* values are relatively constant Fig. 7.6, which is due to the fact the segment consists of normal respiration.

When it comes to detecting events, the Event_{m2} method detected all three events in Fig. 7.6 (4th plot from top), including the event overseen by the Event_{m1} method. However, the Event_{m2} method is not completely immune to noise. Notice how the value barely exceeds the 1.8 value which is the lowest of the four used thresholds in the Event_{m2} method. This caused the Event_{m2} to miss some events, even though the majority of noisy *apneic* events had less noise than the one in Fig. 7.6.

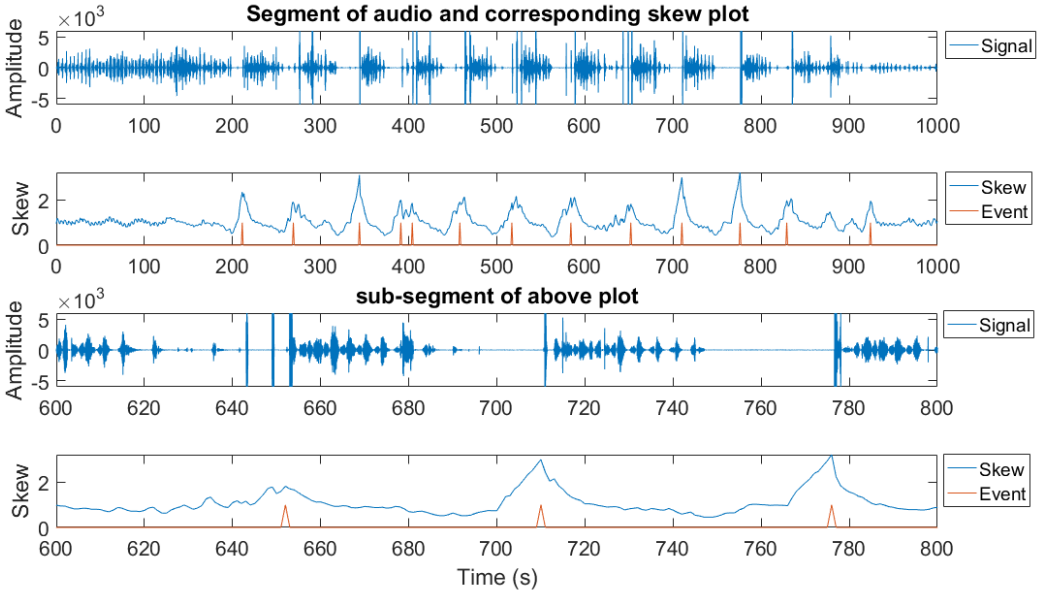


Fig. 7.6: Illustration of how the event_{m2} detection method works. Top plot represents an audio segment from patient 32. Second plot shows the corresponding Sig_{skew} values. The red vertical lines represents instances where the Sig_{skew} values exceeded the threshold, T = 1.8. The third and fourth plot are there for illustrative purpose, and are a zoomed versions of the first two plots.

7.2.3 Hypapnea Event Detection

Fig. 7.7 illustrates two *hypapneic events* with the corresponding parameter values *pAvg* and *Skew* used by Event_{m1} and Event_{m2} to detect events. This is in order to enlighten some of the problems associated with detection of *hypapneic events*

The difficulties of *hypapneic events* using Event_{m1} method, is that the *pAvg* values does not drop sufficiently during the event, to be distinguished from normal breathing with low amplitude. A threshold capable of detecting these events, might also label sections where the subject breathes lightly and mostly out of the mouth, as an event. It is much easier to see the events in the *Skew* values. However, the highest values does not even exceed 1.5, which is below the minimum threshold set, T = 1.8.

Another central issue is that a decrease in airflow, can not be the sole reason for the presence of a *hypapneic*. A lowered *SpO2* values is necessary. Therefore, even if the two signals could become sensible enough to detect *hypapneic* events, they would most likely label segments as *hypapneic* event when they are not.

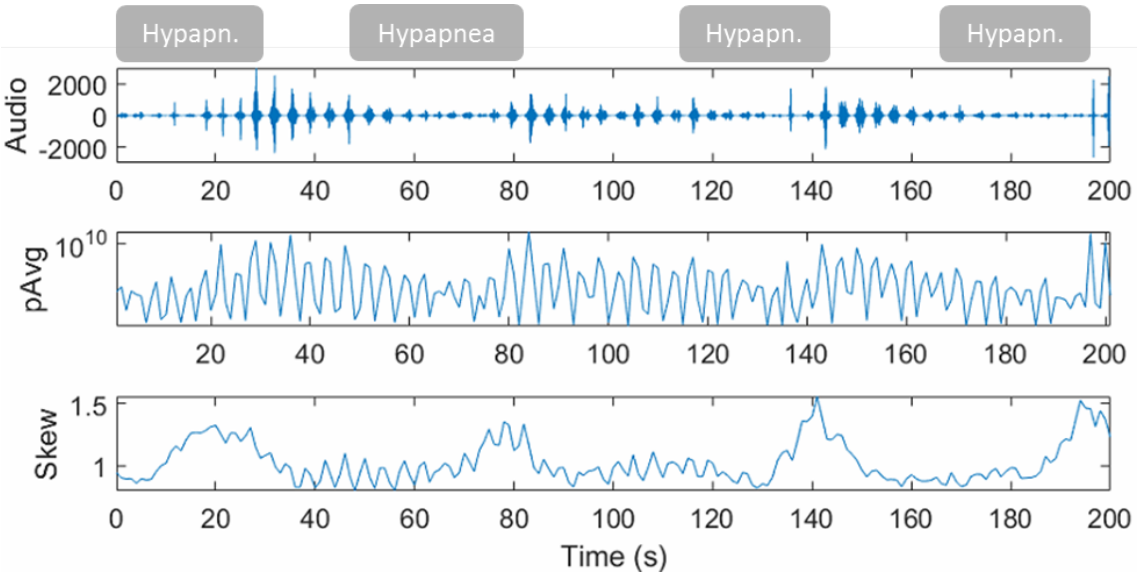


Fig. 7.7: Visualization of problems associated with correct labeling of *hypapneic* events, using segment of the audio signal acquired from Pat 15 (top plot), and the corresponding *pAvg* values (middle plot) and *Skew*-values (bottom plot). Signal sections below the gray boxes are labeled events.

7.2.4 Correlation Between Features and AHI

Below in Fig. 7.8 is seen a correlation plot between all features and the AHI labels. This provides an overview of the variations within each feature and how well it estimates the AHI labels of each subject. The features have all been normalized to better compare the different subplots, opposite to the AHI values which have remained unchanged.

One thing that Fig. 7.8 shows, is that the risk factors; BMI, Agi and Q severity, do not correlate well with the AHI labels. Another thing worth noticing is that Event_{m2} achieves the best correlation, noticeably better than the Event_{m1}. The *surpine* % achieves the worst correlation, which is surprisingly since lying in surpine position, is known to increase AHI.

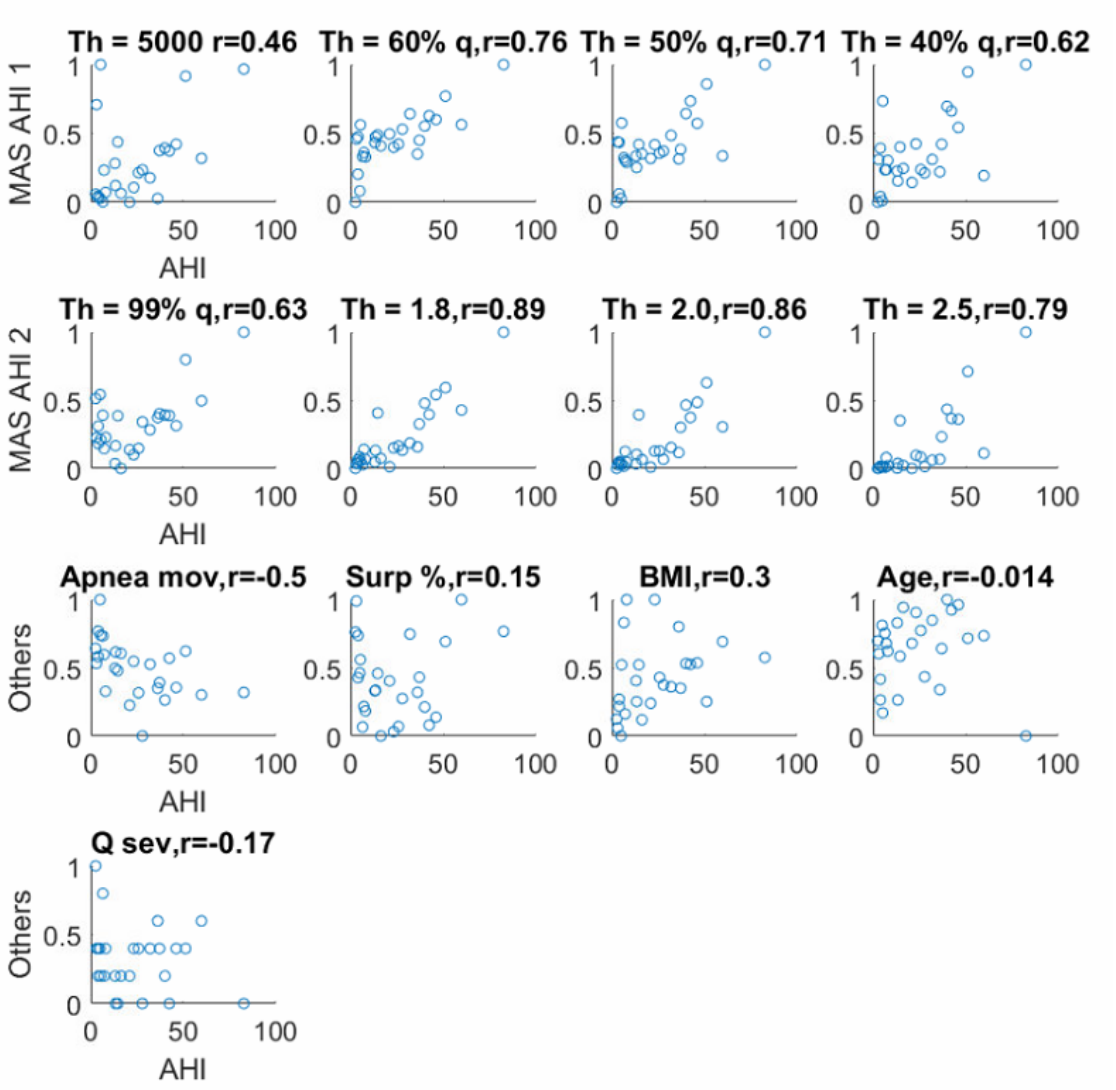
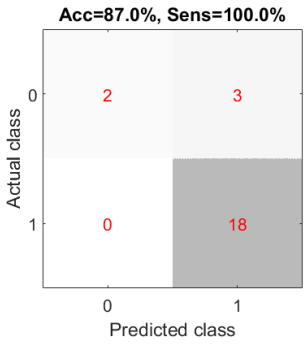


Fig. 7.8: Shows the correlation between the features of each subject related to their AHI-index. Features and their respective correlation coefficient, r, with the feature name is seen in each corresponding subplot title. Notice that the y-axis have not been normalized like the x-axis, in order for it to be easier to compare feature amplitudes with AHI values.

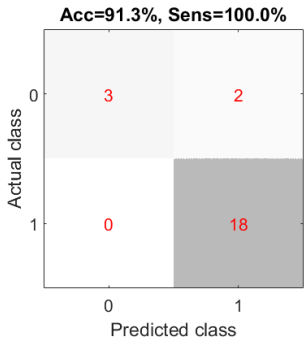
7.3 Classification Performance

This section visualizes the performance of the classifiers using the *confusion matrix* and the performance measurements *accuracy* and *sensitivity*.

Binary Classification, AHI, T = 5



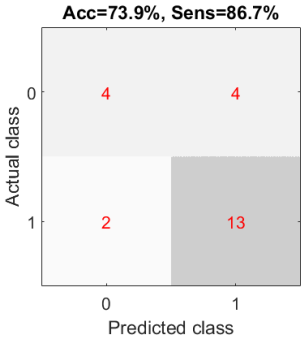
(a) Confusion Matrix of the BDT binary classification, with AHI threshold = 5.



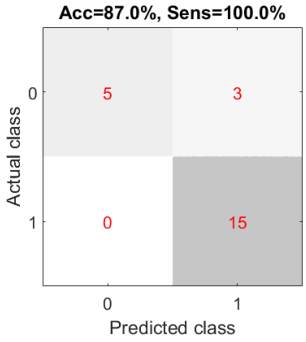
(b) Confusion Matrix of the SVM binary classification, with AHI threshold = 5.

Fig. 7.9: Confusion matrix and performance measures of the two classifiers, *BDT* and *SVM* in the binary classification problem, with $\text{AHI}_{\text{thresh}} = 5$.

Binary Classification, AHI, T = 10



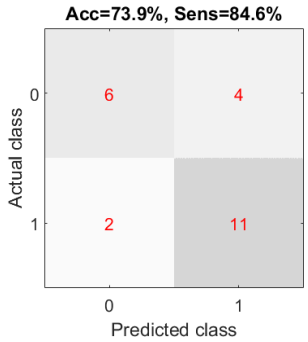
(a) Confusion Matrix of the BTD binary classification, with AHI threshold = 10.



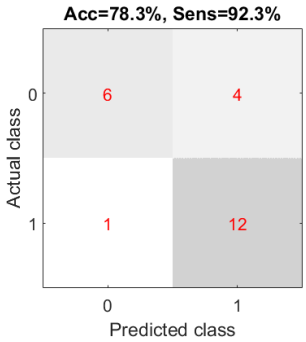
(b) Confusion Matrix of the SVM binary classification, with AHI threshold = 10.

Fig. 7.10: Confusion matrix and performance measures of the two classifiers, *BDT* and *SVM* in the binary classification problem, with $\text{AHI}_{\text{thresh}} = 10$.

Binary Classification, AHI, T = 15



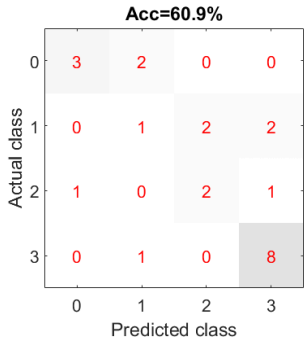
(a) Confusion Matrix of the BDT binary classification, with AHI threshold = 15.



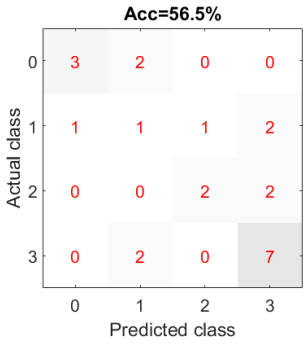
(b) Confusion Matrix of the SVM binary classification, with AHI threshold = 15.

Fig. 7.11: Confusion matrix and performance measures of the two classifiers, *BDT* and *SVM* in the binary classification problem, with $\text{AHI}_{\text{thresh}} = 5$.

Multiclass Classification



(a) Confusion Matrix of the BDT multiclass classification. The sensitivity of each SOA severity class is; [*mild*,*mod*,*sev*] = [20%,25%,89%].



(b) Confusion Matrix of the SVM multiclass classification. The sensitivity of each SOA severity class is; [*mild*,*mod*,*sev*] = [20%,50%,78%].

Fig. 7.12: Confusion matrix and performance measures of the two classifiers, *BDT* and *SVM* in the 4-class classification problem.

Notice that the SVM performs slightly better than the BDT in all of the binary classification problems, achieving sensitivities between 92.3 % and 100 %, as well as accuracies between 78.3 % to 91.3 %. It is clear that both classifiers tend to label more healthy people as having OSA, when the threshold increases, pointing at the problem of separating milder forms of OSA. Fig. 7.12b illustrates this well, since 9 out of 13 missclassifications are from subjects with *mild* or *moderate* OSA. Some interesting findings can be seen by studying the prediction vectors from the multiclass classification process. They prediction vectors are shown in Table. 7.2. Additionally, a table with feature means has been provided in Table 7.3 and is used for analyzing the results in Chapter 8. For a more comprehensive table with all feature values, means and std's, see appendix D.3.1).

ID	1	2	3	4	5	6	10	12	13	14	15	18
BDT	2	3	1	2	3	3	2	3	0	0	1	0
SVM	2	3	1	3	3	3	3	3	3	0	1	0
Lab	1	3	3	2	1	3	1	3	2	0	1	0

ID	19	22	23	24	25	26	29	30	31	32	33	
BDT	1	3	3	3	1	3	3	0	2	3	3	
SVM	1	3	1	2	1	3	3	0	2	3	0	
Lab	0	3	3	2	0	3	3	0	2	3	1	

Table 7.2: Comparison of the predictions made by the BDT and SVM classifier, in relation to the actual labels. Multiclass classification problem. This table will be used in Chapter 8, for discussion classification errors.

Feature	Feat 1	Feat 2	Feat 3	Feat 4	Feat 5	Feat 6	Feat 7
μ Healthy (5)	6.36	17.62	11.08	7.03	11.21	4.92	2.54
μ Mild (4)	13.79	29.17	21.66	16.21	9.54	6.71	3.90
μ Mod (5)	3.50	29.77	20.38	11.98	8.49	9.42	6.18
μ Sev (9)	16.02	41.84	33.13	24.93	12.68	38.39	30.76

Feature	Feat 8	Feat 9	Feat 10	Feat 11	Feat 12	Feat 13
μ Healthy (5)	0.46	1647	0.67	23.2	51.6	3.4
μ Mild (4)	1.36	1579	0.29	30.48	54.25	2.75
μ Mod (5)	2.69	1343	0.13	29.78	65.75	2.5
μ Sev (9)	19.80	1329	0.48	31.2	58.3	2.67

Table 7.3: Table containing mean values for event-features Event_{m1} and Event_{m2} in each severity class. The numbers after each severity class, represents numbers of subjects inside that class.

The trend seen related to how well the classifiers, distinguish subjects with *severe* OSA from the others, makes sense when the mean feature values are seen for the corresponding classes in Table 7.3. For example, notice how features 1 to 8, all have factor two in difference, when comparing *mild* OSA subjects with *severe* OSA subjects. However, some subjects, such as Pat 3, is still classified as *mild* OSA, due to having [*feat 1*, *feat 2*] = [24,17] (see Table in appendix D.3.1) which corresponds to *mild* or *moderate* OSA on Table 7.3.

8 Discussion

8.1 MAS - Data Acquisition

This section discuss the overall procedure of acquiring data using the proposed method. It will contain general aspects of setting the experimental equipment up, as well as different comments about the acquired signals.

8.1.1 Issues Related to Parallel Recording of Signals

The proposed method and the CRM shares some of the same sensor locations (see Fig. 2.3b and Fig 5.1), and therefore some sensors had to be moved to alternative locations. Since the main priority was to be sure that the MAS data acquisition, did not intervene with the CRM sleep study, the placement of the sensors from the CRM had priority. The solution was to put the microphone below the *airflow transducer*, as close to it the location according to the experimental design. A lot of test subjects expressed difficulties in placing it optimally, especially men with beard, since the microphone was very unstable on top of it. Another problem was the size of the smartphone in the proposed method, since the experimental design had it located close to the *signal storage unit* of the CRM and on top of the upper abdomen respiratory belt. This caused the phone to move a lot during the night, some times colliding with the *signal storage unit*. This resulted in noise in the accerometer signal, reducing it's quality. This might be the reason for some inconsistencies in the accelerometer data. Therefore, testing alternative microphone locations or an optimal way of fixating the smartphone in parallel recording might result in better signal qualities.

8.1.2 Acquired Signal Quality

Reduced signal quality in the acquired signals is not only related to problematic sensor placement. Opposite to the other solutions reviewed in the state of the art chapter, the data acquisition in the proposed solution is solely carried out by the subject in their home. Therefore, in general there were two reasons for signal quality reduction; poor placement of sensors, and noise associated with the environment of a private home. Of these two, the biggest issue was by far the sensor placement. As previously illustrated in Fig. 7.2 it could result in a lot of noise in the accelerometer signal. The biggest issue related to the acquired audio signal, was the variation of general audio signal amplitude across subjects while breathing normally. Some had sections of very low audio amplitude, without any signs of *hypapneic* event in the CRM analysis. This might be due to a lot of reasons such as poor placement of the microphone, the subject is breathing through the mouth, the subject is breathing very lightly and so on. Snoring easily detected in the acquired audio signal, due to the high audio amplitude. This caused breathing patters to be very clear when the subject was snoring, but it also proves that a snoring bed partner might be critical for the recordings. One solution could be recording audio with the build in microphone

in the smartphone parallel with the extern one used in the proposed method, and then filter out the snoring partner using *spatial filtering*.

8.1.3 Comparing CRM and MAS Analysis

Synchronizing the recordings of the CRM and the proposed method was important, in order to do the classification on the same recorded segment, and to visualize *labeled segments* of the proposed method. The synchronization of the signals was solved using *time stamps* in each recorded signal, which proved to provide an exact synchronization. The method of extracting the relevant time segment as explained in Fig. 6.2, only worked properly when the MAS recordings started prior to the CRM and ended after the CRM recordings. Some instances were found, where events were annotated in the CRM data, after the CRM recordings had ended. This would cause the AHI of the MAS to be calculated based over a longer time span and thus become lower that it should have been. Correcting this inconsistency, could be done by adjusting the AHI features based on the time length differences between the CRM and the MAS recording, which could improve performance.

8.2 MAS - Advanced OSA Screening Algorithm

This section discuss the performance of the novel OSA screening algorithm developed as part of the system. The results of the screening algorithm exceeded all goals stated in the *Project Goals* section, with the exception of the multiclass problem. The pros and cons of the proposed method will be discussed in the following subsection.

8.2.1 Respiration and Heart Rate estimation

Even though the RR had promising results in the initial experiments on the author, friends and family members (see Appendix D.1), the acquired data in the experiment did not.
The general trend was as seen in Fig. 7.3, which was concluded to be too noisy for assessment for use in the feature set. This could be due to the before mentioned poor fixation of the sensors or the problems associated with parallel conduction of the proposed system together with the CRM. It could also be the due to another problem associated with sleeping in right or left lateral sleeping position. This position tends to cause elastic bands (such as the modified armband) strapped around the chest to loosen. This too, is a problem with the respiration belts, used in the PSG and CRM. The HR estimation was as anticipated, not successful (see Fig. 7.3), since it requires the smartphone to have close contact with the body, which was as previously mentioned, not the case in the proposed solution.

Due to the questionable performance of the RR and HR estimation algorithms, no features were extracted from them. They should be validated before being used; the HR with the recorded HR of the CRM and the RR with the RR extracted from the nasal air transducer. This could open up for interesting opportunities, such as the correlation between Heart Rate Variability (HRV) and OSA severity, found by Gong X. et al. [40] and Park DH. et al. [41].

8.2.2 Inconsistencies in CRM Scoring

One of the core features used in the MAS algorithm for assessing OSA severity was the event finding features Event_{m1} and Event_{m2}. The initial idea was to create an algorithm capable of finding *apneic* and *hypapneic* events, and then train a machine learning algorithm from features extracted

in each event. However, studies have been done related to the accuracy of PSG scorings between experienced scorers. One of these studies was done by Magalang U. J. et al [42], who tested scorings of 15 PSG recordings from experienced sleep scorers, spread across 9 sleep centers. The interscore agreement of the AHI was high, with a intra-class correlation (ICC) of 0.95 [1] but when it came to *total apneas*, the ICC was only 0.73 [2]. This was due to the problem that in some sleep centers, scorers tended to score events more frequently as *hypapneic* than *apneic*, and the other way around in other centers. It is important to note that just because there are big variations in PSG scorings, it is not neccesarily the same with the CRM. However, it is worth keeping in mind, since the CRM has less parameters, and thus might make it harder to correctly analyze recordings.

This is problematic in machine learning relations due to two things; machine learning algorithms are prone to inconsistencies in data, and *hypapneas* are hard to define without an *oximeter* meaning that the algorithm might not be able to distinguished between *hypapneic* events and normal respiration. The combination of these two problems, and state of the art research, lead to the decision of using the AHI to label subjects into categories of OSA severity and use those labels in the supervised learning.

8.2.3 Event Finding

The process of finding a good solution to locate events in the audio signal lead to two methods; Event_{m1} and Event_{m2}. When it came to detecting *apenic* events, both event detection algorithms performed well, but each with their own limitations.

One limitation of the Event_{m1} was due to small voice segments with the before mentioned *gasping* after air, during an *apenic* event. It often lead to an undetected event (as the case in Fig. 7.5) or lead to an event being split into two events due to the sequence of consecutive *no-breath* segments being broken into two or more segments. This could be corrected by implementing an artifact correction algorithm, that removed these small *gasp* segments. The second event detection method, Event_{m2}, was build on a different foundation, namely the relative changes in signal power over a window. The idea was, that if you took the signal power inside these small *gasps* and calculated it relative to a 10 second window, they were not that significant (see Fig. 7.6). This made the Event_{m2} better at labeling "noisy events", than Event_{m1}. However, some events had multiple peaks in *Skew*-values, and thus they would be scored multiple times. A peak finding algorithm decreased this trend, but a smoothing filter (fx. *moving average*. Also, further analysis of segment and window size in the calculations of the PSD should be conducted as well as a more comprehensive analysis of threshold values for each event feature.

The biggest challenge related to event detection was to label *hypapneic* events. As seen in Fig. 7.7, hypapneic events wont get labeled by Event_{m1}, since it is based on consecutive *no-breath* segments, which has a much larger reduction of airflow than the 30 % which is one of the definitions of the event. Event_{m2} succeeded in labeling some *hypapneic* events, but only those with a large reduction in signal amplitude, possible considerably more than the 30 %. The introduction of an *oximeter* might provide the necessary information to improve the performance of detecting *hypapneic* events.

8.2.4 Feature Correlation with AHI

The event finding algorithms proved to have the best correlation with the AHI, where the robustness of Event_{m2} over Event_{m1} is clear when looking at Fig. 7.8. As mentioned in the

[1] 95% confidence interval
[2] 73 % confidence interval

state of the art, there is a big problem in distinguishing between milder forms of OSA. This is one of the downsides of AHI, since it does not take into account the length or severity of an event. Even though this might not be relevant when it needs to correlate with the AHI, it could prove usefull when assessing OSA severity.

The demographic features, gathered through the questionnaire did not correlate well with the AHI, as seen in Fig. 7.8. Small datasets tend to have less correlation with the risk factors, since they are more prone to outliers. For example, the subject who had the highest AHI index in the study, was also the youngest of all the participants and had an average BMI compared to the dataset. A bigger study would theoretically cause a higher correlation with the risk factors, and thus making them more reliable as features.

The two features gathered from the accelerometer, *Apneic movement* and surpine$_{\%}$, also proved to correlate very poorly with the AHI (-0.5 for *apneic mov* and 0.15 for surpine$_{\%}$). Poor fixation of the smartphone could be the cause of low correlation between *Apneic movement* and the AHI. The reason why the surpine$_{\%}$ correlated so poorly with the AHI, should be further investigated, since this is contradictory with the clinical fact, that surpine sleep position increases the risk of obstructed upper airways.

8.2.5 Classification and Evaluation

In general, the classifiers distinguished subjects with *severe OSA* well from the other severity classes as well as healthy subjects. This is reflected in the mean feature values corresponding to each class (see Table. 7.3). All 8 event finding features have considerably large mean values for *severe OSA* compared to the rest of the classes. This difference between means becomes considerably smaller, between the three other classes; *healthy*, *mild* and *moderate*, giving an explanation of why the classifiers performs worse when separating milder forms of OSA (see table. 8.1). This could also be an explanation of the reduced performance when the threshold gets elevated in the binary class problems. It could be interesting to analyze on the percentage of *hypapenic* events related to the OSA severity. If this proved that less severe types of OSA contained higher percentage of *hypaneic events*, the difficulties associated with separating milder forms of OSA could be related to the difficulties of locating *hypapneic* events with the event finder algorithms.

	Bagged Decision Tree		Support Vector Machine	
Method	**Acc** (%)	**Sens** (%)	**Acc** (%)	**Sens** (%)
Binary, TH = 5	87.0	100	91.3	100
Binary, TH = 10	73.9	86.7	87.0	100
Binary, TH = 15	73.9	84.6	78.3	92.3
4-Class	60.9	mild: 20 mod: 50 sev: 89	56.5	mild: 20 mod: 50 sev: 77.8

Table 8.1: Classification performance using all features. The *inter-class sensitivities* are stated in the 4-Class problem for all three severity classes.

All put together, there is still a lot parameters that can be tweaked in both classifiers. For example using a RBF kernel for the SVM, followed by 2-fold crossvalidation with different parameter values and a grid search for the best set, could improve the performance of the SVM.

The same optimization process could be done with the BDT or it could be changed to a Random Forest (RF), testing for performance and computational complexity.

9 Conclusion

The first task of the project was to create an experimental design capable of acquiring biomedical signals measurements for OSA severity assessment, with specified requirements. In accordance with the project goals, the proposed method used a smartphone as the mainframe of the signal acquisition a long with equipment found in most homes (smartphone and microphone), with the exception of the modified armband. The experimental design was used to perform all-night data acquisition, using just about 20 % power consumption and approx. 1 GB data storage for each recording. This was almost twice as effective as the specifications stated in the Project Goals.

The second task was to create an advanced signal screening algorithm capable of analyzing acquired data and use it to characterize subjects into categories associated with the absence or presence and severity of OSA. The features extracted from the acquired biomedical signals, was based on event detection, movement during sleep and demographic information. The classification was tested with two classifiers, in two different algorithm setups; the SVM and the BDT. Both setups was tested using three binary classification problems and one multiclass. Results of the event finding algorithms suggests that adding an *oximeter* would increase the performance, by increasing the possibility of detecting *hypapneic* events. A more comprehensive analysis of feature parameters, thresholds and alternatives should be investigated to improve results further.

The final task were to validate the before mentioned algorithm. This was done by conducting a CRM sleep study parallel to the proposed method, on a total of 23 subjects. The SVM had the best performance in the binary classification problems reaching *sensitivities* above the project goal of 90 % in all three classification problems and only had one case of *accuracy* below the project goal of 80 %. The 4-Class classification proved to be less successive. As anticipated, the BDT had better classification performance in the 4-class problem than the SVM, achieving 60 % *accuracy* and 89 % *sensitivity* for the *severe* class. However, *mild* OSA achieved a *sensitivity* of only 20 % and *moderate* 50 %.

This project has shown some promising results, performed well compared to state of the art solutions, and succeeded in reaching all of the project goals, except performance goals of the multiclass problem. However, it also showed some limitations regarding the event detection, and the detection of *hypapneic* events. Overall, the project was considered a success as a *proof of concept* solution, providing valuable knowledge to the field of OSA screening.

9.1 Future Work

The field of mobile screening of OSA is a relatively new one, reflected by the big differences between approaches in the state of the art, regarding sensors and features. The study was mainly based on event detection algorithms, which worked great, but with room for improvements. One improvement that would most likely contribute to the performance of the algorithm would be the introduction of an *oximeter*. The HR and RR estimation methods should be further investigated for potential use in the feature set. Furthermore, multiple features could be investigated for potential inclusion in the feature set. Variations of the classifiers should also be tested, using 2-fold crossvalidation and gridsearch to find the optimal parameters.

In order to make it a fully transportable system, where users are capable of performing the full test themselves, the whole signal processing algorithm should be implemented in the Android application. Finally, more experimental tests should be conducted, in order to be able to validate the system to achieve a greater statistical foundation.

10 Bibliografy

[1] Berger, M., Oksenberg, A., Silverberg, D. S., Arons, E., Radwan, H., & Iaina, A. (1997). *Avoiding the supine position during sleep lowers 24 h blood pressure in obstructive sleep apnea (OSA) patients*, Journal of Human Hypertension. 11(10), 657-64.

[2] Eckert, D. J., & Malhotra, A. (2008) *Pathophysiology of Adult Obstructive Sleep Apnea*. Proc Am Thorac Soc. 5(2), 144-153.

[3] Lifescript, webside accessed 1/21-17, url: http://www.lifescript.com/health/a-z/mayo/images/o/obstructive_sleep_apnea.aspx.

[4] Jennum, P. et Kjellberg, J. (2011), *Health, social and economical consequences of sleep-disordered breathing: a controlled national study*, Thorax 66(7), 560-566

[5] Finkel, K. J., Searleman, A. C., Tymkew, H., Tanaka, C. Y., Saager, L., Safer-Zadeh, E., ... Avidan, M. S. (2009). *Prevalence of undiagnosed obstructive sleep apnea among adult surgical patients in an academic medical center*, Sleep Medicine, 10(7), 753-758.

[6] Bucklin, C. L., Das, M., & Luo, S. L. (2010). *An Inexpensive Accelerometer-Based Sleep-Apnea Screening Technique*, Naecon, 396-399.

[7] Sleep: A Comprehensive Handbook,SLEEP(2006). *Sleep: A Comprehensive Handbook,SLEEP*, John Wiley & Sons Inc.

[8] Mayo Foundation for Medical Education and Research,*Obstructive Sleep Apnea image*, url: http://sleeptoliveinstitute.com/disorders-diagnostics/sleep-apnea-snoring, Date: 22/10 2016

[9] Jerome M. Dempsey, Sigrid C. Veasey, Barbara J. Morgan, and Christopher P. O'Donnel (2010). *Pathophysiology of Sleep Apnea*, 48-93.

[10] Malhotra, R. K. (2015). *Sleepy or Sleepless: Clinical Approach to the Sleep Patient*. Springer International Publishing.

[11] Righospitalet Glostrup. *Polysomnografi (PSG) med video under indlæggelse*. Rigshospitalet Glostrup, Neurofysiologisk Afdeling Maj 2016.

[12] Righospitalet Glostrup. *CRM udført i eget hjem - Igangværende patienter*. Rigshospitalet Glostrup, Neurofysiologisk Afdeling Maj 2016.

[13] Berry, R. B., Budhiraja, R., Gottlieb, D. J., Gozal, D., Iber, C., Kapur, V. K., ... Tangredi, M. M. (2012). *Rules for Scoring Respiratory Events in Sleep: Update of the 2007 AASM Manual for the Scoring of Sleep and Associated Events*. Journal of Sleep Medicine, 8(5), 597-619.

[14] Basner, R. C., Ringler, J., Schwartzstein, T. M., Weinberger, S. E., & Weiss, J. W. (1991). *Phasic electromyographic activity of the genioglossus increases in normals during slow-wave sleep.*. Respiration Physiology, 83(2), 189-200.

[15] Corso, R. M., Gregoretti, C., Braghiroli, A., Fanfulla, F., & Insalaco, G. (2015). *Correlation between the STOP-BANG score and the Severity of Obstructive Sleep Apnea*. Anesthesiology, 122(6), 1437-1438.

[16] Block K.E. (1997). *Polysomnography: a systematic review*,Technology and Health Care, 5(4), 285-305.

[17] Nagappa, M., Liao, P., Wong, J., Auckley, D., Ramachandran, S. K., Memtsoudis, S., ... Chung, F. (2015). *Validation of the STOP-Bang Questionnaire as a Screening Tool for Obstructive Sleep Apnea among Different Populations: A Systematic Review and Meta-Analysis*, Plos One, 10(12).

[18] Nakano, H., Hirayama, K., Sadamitsu, Y., Toshimitsu, A., Fujita, H., Shin, S., & Tanigawa, T. (2014). *Monitoring Sound To Quantify Snoring and Sleep Apnea Severity Using a Smartphone: Proof of Concept*, Journal of Clinical Sleep Medicine, 10(1), 73-78.

[19] Al-Mardini, M., Aloul, F., Sagahyroon, A., & Al-Husseini, L. (2014). *Classifying obstructive sleep apnea using smartphones*, Journal of Biomedical Informatics, 52, 251-259.

[20] Behar, J., Roebuck, A., Shahid, M., Daly, J., Hallack, A., Palmius, N., ... Clifford, G. D. (2015). *SleepAp: An Automated Obstructive Sleep Apnoea Screening Application for Smartphones*, IEEE Journal of Biomedical and Health Informatics, 19(1), 325-331.

[21] Oliver, N., & Flores-Mangas, F. (2007). *HealthGear: Automatic Sleep Apnea Detection and Monitoring with a Mobile Phone*, Journal of Communications 2(2), 1-9.

[22] Alqassim, S., Ganesh, M., Khoja, S., Zaidi, M., Aloul, F., & Sagahyroon, A. (2013). *Sleep Apnea Monitoring Using Mobile Phones*, IEEE 14th International Conference on e-Health Networking, Applications and Services 2012.

[23] Rendón, D. B., Ojeda, J. L. R., Foix, L. F. C., Morillo, D. S., & Fernandez, M. A. (2007). *Mapping the Human Body for Vibratons using an Accelerometer*, EMBS 2007. 29th Annual International Conference of the IEEE.

[24] Sanchez Morillo, D., Rojas Ojeda, J. L., Crespo Foix, L. F., & Leon Jimenez, A. (2010). *An Accelerometer-Based Device for Sleep Apnea Screening*, Ieee Transactions on Information Technology in Biomedicine, 14(2), 491–499.

[25] Shi, C., Nourani, M., Gupta, G., & Tamil, L. (2013). *Apnea MedAssist II: A smart phone based system for sleep apnea assessment*, Bioinformatics and Biomedicine, IEEE International Conference 2013, 572-577.

[26] Berry, R. B., Budhiraja, R., Gottlieb, D. J., Gozal, D., Iber, C., Kapur, V. K., ... Tangredi, M. M. (2012). *Rules for Scoring Respiratory Events in Sleep: Update of the 2007 AASM Manual for the Scoring of Sleep and Associated Events*, Journal of Clinical Sleep Medicine, 8(5), 597–619

[27] Sola-Soler, J., Antonio Fiz, J., Morera, J., & Jane, R. (2012). *Multiclass classification of subjects with sleep apnoea-hypopnoea syndrome through snoring analysis.* Medical Engineering and Physics, 34(9), 1213–1220.

[28] Halevi, M., Dafna, E., Tarasiuk, A., & Zigel, Y. (2016, August). *Can we discriminate between apnea and hypapnea using audio signals*, In Engineering in Medicine and Biology Society (EMBC), 2016 IEEE 38th Annual International Conference of the (pp. 3211-3214). IEEE.

[29] www.gsmarena.com. Accessed 26-09-2016,11:33.

[30] www.stackoverflow.com. Accessed 26-09-2016,11:35.

[31] url: http://labinyourpocket.com/sensors-the-accelerometer/

[32] *Labinyourpocket*, url: http://labinyourpocket.com/sensors-the-accelerometer/, last accessed; 10/21-2016

[33] Rangaraj R. M. (2002) *Biomedical Signal Analysis: A Case-Study Approach*.

[34] Bates, A., Ling, M. J., Mann, J., & Arvind, D. K. (2010). *Respiratory rate and flow waveform estimation from tri-axial accelerometer data*. 2010 International Conference on Body Sensor Networks, Bsn 2010, 5504743, 144–150.

[35] Park, M., & Gao, Y. (2006). *Error Analysis and Stochastic Modeling of Low-cost MEMS Accelerometer*. Journal of Intelligent and Robotic Systems, 46(1), 27–41.

[36] Biau, G., & Scornet, E. (2016). *A Random Forest Guided Tour*, Test, 25(2), 197–227.

[37] Tan, P.-N., Steinbach, M., & Kumar, V. (2014). *Introduction to Data Mining*, 1st Edition, Pearson Education Limited 2014.

[38] Bishop C. M. (2006).*Pattern Recognition and Machine Learning*, 1st Edition, Springer 2006.

[39] Winston P. H,*6.034 Artificial Intelligence*, MIT Lecture Fall 2010, Source url: www.ocw.mit.edu.

[40] Gong, X., Li, C., Mao, X., Liu, W., Huang, X., Chu, H., ... Lu, J. (2016). *Correlation Analysis between Polysomnography Diagnostic Indices and Heart Rate Variability Parameters among Patients with Obstructive Sleep Apnea Hypopnea Syndrome*, Plos One 2016;1-13

[41] Park, D.-H., Shin, C.-J., Hong, S.-C., Yu, J., Ryu, S.-H., Kim, E.-J., ... Shin, B.-H. (2008). *Correlation between the severity of obstructive sleep apnea and heart rate variability indices*, Journal of Korean Medical Science, 23(2), 226–231.

[42] Magalang, U. J., Chen, N.-H., Cistulli, P. A., Fedson, A. C., Gslason, T., Hillman, D., ... Investigators, S. A. G. I. C. (2013). *Agreement in the Scoring of Respiratory Events and Sleep Among International Sleep Centers*. Sleep, 36(4), 591–596.

A Appendix: Conference Paper

This appendix includes the Conferance Paper which will be submitted to the 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, after the thesis has been handed in.

Mobile Apnea Screening System for *at-home* Recording and Analysis of Sleep Apnea severity.

Mathias P. Bonnesen[1], Poul Jennum[2] and Helge B.D. Sorensen [3]

***Abstract*— Obstructive Sleep Apnea (OSA) is a common sleep disorder estimated to affect 2-4 % of the middle-aged population. The golden standard diagnostic procedure is the full Polysomnography (PSG), which is both costly and time consuming to perform. Therefore a simple and cheap alternative would be of great value.**

This study presents a novel *at-home* screening method for OSA using a smartphone, a microphone and a modified armband, to measure continuous biological signals during a whole nights sleep. A signal processing algorithm is used to classify the subjects, into classes according to severity of the disorder. The system was validated by conducting a sleep study parallel to the data acquisition on a total of 23 subjects.

Both binary and 4-class classification problems was performed. The results of the binary classifications had performed the best with *sensitivities* between 92.3 % and 100 %, and accuracies between 78.3 % and 91.3 %. The multi-class classification was not as successful with *sensitivities* of 75 %, and *accuracies* of 56.5 % and 60 %). The primary cause of reduced classification performance, was found to be the difficulty of distinguishing between milder forms of OSA.

I. INTRODUCTION

Obstructive Sleep Apnea (OSA) is a common sleep disorder estimated to affect 2-4 % of the adult population, with a variation across ethnic groups. The disease is characterized by events of reduced or seized airflow due to an obstruction in the the upper airways, and the severity is based on the frequency of these events [1]. Complications related to the disorder ranges from sleepiness and decreased learning ability to more severe complications such as depression and increased risk of getting cardiovascular diseases. An estimated 90 % of OSA sufferers are undiagnosed [1], causing huge social, health and economical consequences to the society [2] due to reduced quality of life for a lot of these individuals. Therefore this is a problem that should be addressed.

The current gold standard diagnostic method is the polysomnography (PSG), which is both costly and time consuming. It is usually recorded in a sleep clinic over night, and then scored by medical professionals, into an event-rate parameter, called the Apnea-Hypapnea Index (AHI). Recent developments of OSA diagnostics have been the introduction of the Cardio-Respiratory monitor (CRM), allowing simpler and *at-home* recording of biological signal parameters. However, it is still an expensive medical device which needs health professionals

These issues, combined with the fact that an estimated 50 % of subjects referred to sleep analysis at Glostrup Rigshospital are found to be healthy, results in huge medical expenses that could have been spent elsewhere. Multiple researchers have tried to address this issue like Al-Mardini M. et al. [7] who used event detection from an audio signal and SPO_2 from an oximeter, to classify subjects into severity of OSA. Others, such as Behar J. et al. [5] analyzed on general trends in audio and accelerometer signals together with *SpO2* event detection for classifying subjects into OSA positive or OSA negative.

This study focuses on implementing a simple and easy-to-perform, portable solution for extracting biological signals over the duration of a full nights sleep, in order to screen for OSA. The experimental design was limited to only include a smartphone, a microphone and a modified armband. The smartphone was both used as a main-frame for all signal acquisition, and to gather demographic information, by using a STOP-BANG questionnaire.

The signal processing algorithm uses modified event-detection algorithms from Al-Mardini M. et al. [7] and Halevi M.J et al. [6], to make them more robust to noise. Features related to movement and sleep position, as well as demographic info is also added to the training set.

II. MATERIALS AND METHODS

A. Description of Data

The dataset acquired in the experiments included 23 subjects; 17 men and 5 women. Table I below, shows statistics related to the data.

TABLE I
DATA STATISTICS

	Mean	Std	Min	Max
BMI	29.4	5.3	20.4	41.4
Age	57.4	14.3	22.0	75.0
AHI	26.2	21.4	2.6	82.6

Of the 23 test subjects, 5 were healthy, 4 had *mild* OSA, 5 had *moderate* OSA and 9 had *severe* OSA, using the multiclass classification thresholds from the state-of-the-art.

[1]Mathias P. Bonnesen is with the Department of Electrical Engineering, Technical University of Denmark

[2]Chief Physician, Professor, M.D. P. Jennum with the Danish Center for Sleep Medicine, Department of Clinical Neurophysiology, Glostrup University Hospital, Glostrup, Denmark.

[3]Associate Professor, M.Sc.E.E., Ph.D H. B.D. Sorensen with the Derpartment of Electrical Engineering, Technical University of Denmark, Lyngby, Denmark.

B. Experimental Setup

The data was collected at the *Danish Center for Sleep Medicine, Department of Clinical Neurophysiology*, on subjects referred to the clinic for a CRM sleep study. In order to be able to validate the system, parallel CRM was recorded, and scored by the medical staff. An LG Nexus 5 smartphone, with a build-in tri-axis accelerometer was placed on *suprasternum* with the screen facing outwards and jack-stick input facing upwards. The modified armband was used for fixation purposes, and contained the smartphone. The microphone was placed next to the nose, on line with the nostrils, see Fig. 1.

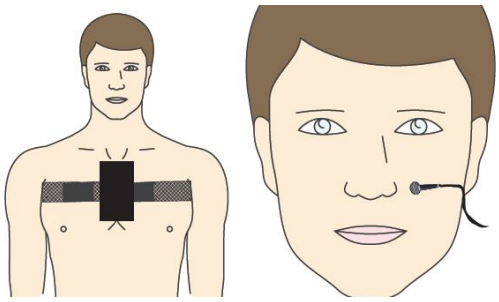


Fig. 1. Experimental setup. The correct placement of the experimental setup; smartphone inside the modified armband (left) and the microphone (right).

C. Data acquisition

1) Platform: The framework for the data-acquisition was done on a LG Nexus 5 using android IOS 6.0.1, which sampled the signals continuously over the duration of the recording, and saved them in .txt files. One full-night recording (6-8 hours) consumed approximately 20-25 % battery power and took approximate. 1 gb of storage.

2) Android application: The android application was created to be as simple as possible, using only two activities; one for the implemented STOP-BANG questionnaire and one for handling the signal acquisition. The audio sampling was done using a sampling frequency of 16 kHz and the accelerometer data was sampled with a sampling frequency of 50 Hz.

D. Feature extraction

The feature extraction was done in MATLAB v. R2016a, where the acquired signals was initially down-sampled. The audio was down-sampled to 4 kHz and the accelerometer was re-sampled, to account for the sample frequency inconsistency down to 25 Hz. The audio was filtered using a IIR bandpass filter with cutoff frequencies between 200 Hz and 1200 Hz, and the accelerometer signals were filtered using an IIR bandpass filter with cutoff frequencies between 0.1 Hz and 0.8 Hz..

1) Event detection, - Method 1: The first even detection method, $Event_{m1}$ is a modified version of the one from Al-Mardini et al. [7], in that the proposed method used the *Average Power* (*pAvg*) of the signal and not the signal energy. The first step was therefore to calculate the Power Spectral Density (PSD) of the audio signal, using *pwelch* method [8]. The next step was to calculate the average power in 1 second windows, and label the segment as *breath* or *no-breath* depending on a specified threshold value. An event was defined as 10 consecutive *no-breath* windows. The final step was to calculate the event rate, using Eq. 1

$$Event_{m1} = \frac{event_{tot}}{Length_h} \tag{1}$$

, where $event_{tot}$ is the total events and $Length_h$ is the signal length, in hours. Four different thresholds ware used; 500 and three quantiles of the whole segment; 60 %, 50 %, 40 %, resulting in four features (see Table III)

2) Event detection, - Method 2: The second event-detection algorithm $Event_{m2}$ is a modified version of Halevi M.J. et al. [6], in that it analyzes the total power in a window segment before and after a given sample, calculating the power difference between the two. The *pAvg* of each window segment was extracted the same way as in the $Event_{m1}$. The relative value before and after the sample was calculated using Eq. 2.

$$Skew(n) = \frac{\sum_{i=1}^{N} log(x_{n+i})}{\sum_{j=1}^{M} log(x_{n-j})} \tag{2}$$

, where *n* is the sample number, *i* is the window of samples after sample *n*, and *j* is number of samples before *n*. Eq. 2 uses the logarithmic value of *pAvg* in order to avoid large skewness values from high-pitch noise segments. Afterwards values exceeding a certain threshold is extracted from the $Skew$, and the vector locations are defined as *events*. The even rate feature $Event_{m2}$ is calculated using Eq. 1. Four different thresholds was used; 99 % quantile and 1.8, 2, 2.5, resulting in four features (see Table. III)

Fig. 2 provides an insight in how *apneic events* were reflected in the *pAvg* values and the *Skew* values.

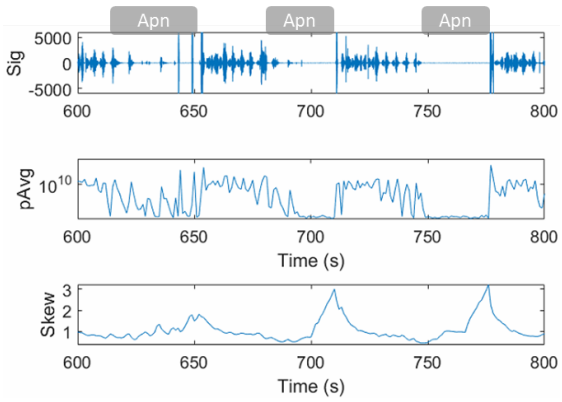


Fig. 2. Overview of the event detection processes. The top graph is an audio segment with apneic events, the graph in the middle is the *pAvg* values used in $Event_{m1}$ and the lower graph is *Skew* values used in $Event_{m1}$. Signals under the gray boxes, are *apneic* events

3) Sleep position: The accelerometer readings were used to extracting two different features; body position and movement.

The body position was based on the gravitational influence on the accelerometer. Since the gravitational force, G ≈ 9.8 $\frac{m}{s^2}$, is large compared to activity related to movement while sleeping, a simple set of rules was used to determine the sleep position, as shown in Table II.

TABLE II
RULES ASSOCIATED WITH DETERMINING SLEEP POSITION. ALL VALUES ARE APPROXIMATES

Body position	**Rule 1**	**Rule 2**	**Index**
Surpine	Z > X and Y	Z > 0	1
Prone	Z > X and Y	Z < 0	2
Left Lateral	X > Y and Z	X < 0	3
Right Lateral	X > Y and Z	X > 0	4

4) Apneic Movement: Apneic movement was determined by first filtering all three accelerometer axis using an IIR bandpass filter with cutoff frequencies of 0.1 and 0.8. Then the Signal Vector Magnitude [7] was calculated using Eq. 3, to reflect combined body movement in all axis.

$$c_i = \sqrt{x_i^2 + y_i^2 + z_i^2} \tag{3}$$

The total *apneic movement* was defined as values exceeding a threshold defined as the 50 $\%$ quantile of the whole signal.

5) Questionnaire: The STOP-BANG questionnaire wes used to extract demographic info such as age and BMI, but also to provide a severity value based on the answers. A total of six questions was asked, where a "yes" counted as 1 point and "no" counted as 0 points. A full overview of all signals provided in Table III.

TABLE III
LIST OF THE 13 FEATURES EXTRACTED FROM THE DATA.

Feature	**Source**	**Unit**	**Amount**
Surpine	Acc	$\%$	1
Apnea mov	Acc	Seconds	1
$Event_{m1}$	Aud	Event rate (1/hour)	4
$Event_{m2}$	Aud	Event rate (1/hour)	4
BMI	STOP-BANG	$\frac{kg}{m_2}$	1
Age	STOP-BANG	Years	1
Severity	STOP-BANG	[1,2,...,6]	1

E. Classification

Three binary classification problems and one 4-class classification problem was tested, using two different classifiers; Bagged Decision Tree (BDT) and the Support Vector Machine (SVM). The AHI-thresholds used for the binary problems were 5, 10 and 15. For the multiclass problem the intervals were as seen below

- Healthy: AHI ≤ 5
- Mild: $5 \leq$ AHI ≤ 15
- Moderate: $15 \leq$ AHI ≤ 30
- Severe: AHI ≥ 40

The RF classifier used the purity measure *Gini* and a total of 50 trees in the ensemble. The SVM was implemented with a *linear kernel*, *boxConstrant = 1* and *one-sv-one* Coding for the 4-class classification problem.

F. Validation

The data was scored by the medical staff at the Danish Center for Sleep Medicine, resulting in AHI labels and event labeling (*Apneic* and *hypapneic*). An *apneic* event is defined as a 90 $\%$ reduction of airflow for more than 10 consecutive seconds, and *hypapneic* events corresponds to a 30 $\%$ drop in airflow and a 4 $\%$ decrease in *SpO2* levels, for 10 consecutive seconds [9]. Due to the limited size of the dataset, *leave-one-out* cross-validation was used on the dataset with the performance measures *sensitivity* and *specificity*, to determine the performance.

III. RESULTS

The correlation between the features used, and the AHI labels from the parallel conducted sleep study was calculated, in order to give an idea about how well they correlated with the severity of OSA. Table IV below shows the correlations coefficients, *r* belonging to each feature.

TABLE IV
CORRELATION COEFFICIENT, r, BETWEEN AUDIO FEATURES AND AHI LABELS

Feature	**Correlation Coefficient, r**	**Feature**	**Correlation Coefficient, r**
$Event_{m1,1}$	0.46	$Event_{m2,4}$	0.79
$Event_{m1,2}$	0.76	Apnea mov.	-0.50
$Event_{m1,3}$	0.71	Surpine	0.15
$Event_{m1,4}$	0.62	BMI	0.30
$Event_{m2,1}$	0.63	Age	-0.01
$Event_{m2,2}$	0.89	severity	-0.17
$Event_{m2,3}$	0.86		

The classification results for each of the four classification problems can be seen in Table V, for both classifiers used.

TABLE V
CLASSIFICATION PERFORMANCE USING ALL FEATURES. THE INTER-CLASS SENSITIVITIES ARE STATED IN THE 4-CLASS PROBLEM.

	Bagged Decision Tree		Support Vector Machine	
Method	**Acc** ($\%$)	**Sens** ($\%$)	**Acc** ($\%$)	**Sens** ($\%$)
Binary, TH = 5	87.0	100	91.3	100
Binary, TH = 10	73.9	86.7	87.0	100
Binary, TH = 15	73.9	84.6	78.3	92.3
4-Class	60.9	75	56.5	75

As seen in Table V, both classifiers performed well, with SVM achieving slightly better *sensitivities*, compared to the BDT, but relatively equal *accuracies*. This might point to the

fact that the BDT was more likely to classify OSA suffering subjects as healthy, where the SVM where more likely to classify healthy subjects as having OSA.

IV. Discussion

The event finding algorithms provided the best correlation with the AHI-index, especially the $Event_{m2}$, which had a correlation constant *r = 0.89* with threhold 2 and *r = 0.86* with threshold 1 (see Table IV). The reason that $Event_{m2}$ had a higher correlation with the AHI compared to $Event_{m1}$, is that it was less prone to noise under *apneic events*. The noise had to have a large amplitude in order to provide a *Skew* value big enough for an $Event_{m2}$ detection, where $Event_{m1}$ depended on 10 consecutive segments with relatively little noise. To get a better understanding of the strength and weakness of the event-detection algorithms, the values *pAvg* and *Skew* was examined in segments with annotated *apneic events* (see Fig. 2). It illustrates some problems associated with *apneic* events, such as artefacts completely disrupting the *pAvg* values used in $Event_{m2}$ and reducing the *Skew values* in $Event_{m1}$. Removing these artifacts with an artifact-reduction algorithm, would most likely increase performance for both event-detection algorithms.

A much bigger issue was related to the detection of *hypapneic* events. The small decrease in airflow was hard to distinguish from normal breath, since the amplitude varied a lot between subjects. The implementation of an oximeter would likely increase the performance of the event detection algorithms correctly labeling a *hypapneic* events.

Features extracted from the questionnaire did not correlate well with the AHI, with *BMI* achieving the highest correlation of r = 0.3. Even though BMI and Age are known risk factors of OSA, it is important to note that with a dataset of only 23 subjects, the odds of participants following the general trends are not great.

The binary classification went well, despite of highly skewed classes; a $AHI_{thres} = 5$ resulted in 5 out of 23 being OSA positive, $AHI_{thres} = 10$ resulted in 8 out of 25, and $AHI_{thres} = 15$ in 10 out of 25. The high percentege of OSA sufferers in the dataset might reduce the accuracy, but this will also increase *sensitivity*, which could be the explanation of the high *sensitivities*. The 4-class classification problem received significantly lower accuracies than the binary problems. Some evidence of why this is the case, can be seen in Fig. 3. Notice that 9 out of 13 miss-classifications are from subjects with *mild* or *moderate* OSA. Only 1 out of 5 with mild OSA are correctly classified. This points, once again, to the problem of distinguishing between milder forms of OSA.

V. Conclusion

The study succeeded in developing a portable device for *at-home* screening of OSA. The SVM classifier had the best performance in all three binary classification problems reaching *sensitivities* between 85.7 % and 100 %, and accuracies between 78.3 % and 91.3 %. The multi-class classification was with *sensitivities* between 75.0 % and 85.7 %, accuracies between 56.5 % and 60.9 %, primarily due to difficulties associated with distinguishing between milder forms of OSA. This is also seen in the binary classification results, as the performance reduces the higher the threshold i set.

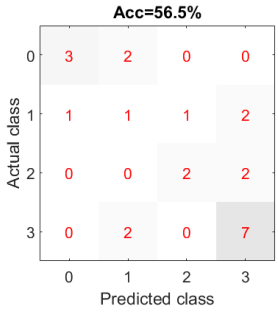


Fig. 3. Confusion matrix of the results from the 4-class classification using the SVM. Correctly labeled subjects are the ones in the diagonal.

There are still a lot of work that needs to be done before it can be commercialized, but as a *proof of concept*, the project was considered a success.

VI. Future work

The addition of an *oximeter* could prove to improve a lot of the problems associated with detecting *hypapneic events*, and thus improve performance considerably. Furthermore, in a future release the signal processing algorithm should be part of the android application, in order to make it fully automatic.

ACKNOWLEDGMENT

References

[1] Berger M. et al.,*Avoiding the supine position during sleep lowers 24 h blood pressure in obstructive sleep apnea (OSA) patients*, J Hum Hypertens 1997;11:657-64.

[2] Jennum P. et Kjellberg J.,*Health, social and economical consequences of sleep-disordered breathing: a controlled national study*, Thorax 2011;66:560-566

[3] Oliver N. et Flores-Mangas F.,*HealthGear: Automatic Sleep Apnea Detection and Monitoring with a Mobile Phone*, Journal of Communications 2007;2:1-9.

[4] Alqassim S. et al.,*Sleep Apnea Monitoring Using Mobile Phones*, IEEE 14th International Conference on e-Health Networking, Applications and Services 2012.

[5] Behar J. et al.,*SleepAp: An Automated Obstructive Sleep Apnoea Screening Application for Smartphones*, IEEE Journal of Biomedical and Health Informatics 2015;19:325-331.

[6] Halevi M. J. et al.,*Can we discriminate between apnea and hypapnea using audio signals*, 38th Annual International Conference of the Ieee Engineering in Medicine and Biology Society (embc) 2016, pp. 3211-14.

[7] Al-Mardini M. et al.,*Classifying obstructive sleep apnea using smartphones*, Journal of Biomedical Informatics 2014;52:251-259.

[8] Rangaraj M. Rangayyan *Biomedical Signal Analysis: A Case-Study Approach*. 2002,The Institude of Electrical and Electronics Engineers.

[9] Berry B. R. et al. *Rules for Scoring Respiratory Events in Sleep: Update of the 2007 AASM Manual for the Scoring of Sleep and Associated Events*. Journal of Sleep Medicine 2012;8:597-619

B Appendix: Litterature Search Method

Two state of the art studies was conducted during the project work; A main study and a sub-study. Below is described the specifications of each study, made in order to focus on relevant articles.

B.1 Main study, Method

Main Study	Time of study: "Done between the 24th august to the 18th of september".
Questions	**Question:** "Which mobile sleep apnea assessment solutions currently exists?",
Inclusion Standards	**Publication years: 2005 to present:** Since it is relatively new for phones to be powerful enough for acquirering biological signals and analysis on them, no articles pre-2005 will be included in the study.
	Studies that has it focus on assessment of sleep apnea. To avoid tons of articles related to general analysis of sleep.
	Mobile solution: Since the goal is to design a system using a smartphone.
Exclusion standards	**Studies that acquires the signals on awaken subjects:** The focus is set on sleep studies
	Studies that does not include one of the three sensors in its analysis; An accelerometer, a microphone or an oximeter: The focus is on sensors that non-medical proffesionals can use.
Search Engines	DTU findit, Web of Science, Google Scholar
Keywords used	Smartphone, sleep apnea, mobile, mobile application, monitor(ing), audio, accelerometer, obstructive sleep apnea, mHealth, sleep disorders.

- **Identification:** The initial search was done with the above specifications on *WebOfScience* between the 24th and 27th of september, resulting in a total of 24 relevent articles. The second search was conducted on *DTU findit* between the 27th of september and and 6th

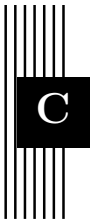
of october, resulting in 11 relevent articles. The final search was conducted on *Google Scholar* between the 8th and the 15th of october, resulting in 26 relevent articles Out of the total of 61 relevent articles found, 22 was dublicates, leaving 39 relevent articles.

- **Screening:** The 39 relevent articles was screened and 5 were excluded due to irrelavence to the project.
- **Eligibility:** For the presentation of current state of the art smartphone solutions an additional exclusion criteria was used namely "Exclude all studies not using smartphones as the main-frame for data acquisition protocol", reducing the amount of articles to 6. The rest of the articles were saved for later use in the Sub-study.
- **Inclusion:** The 6 articles were chosen to be included in the state of the art chapter.

B.2 Sub-study, Method

Sub-study	**Time of study:** "Done between the 4th of october to the 9th of october".
Questions	**Question:** "Which investigations have been conducted related to placement of accelerometer and/or microphone?",
Standards	**Same as Main-Study**
Search Engines	DTU findit, Web of Science, Google Scholar
Keywords used	Smartphone, sleep apnea, mobile, mobile application, monitor(ing), audio, accelerometer, obstructive sleep apnea, mHealth, sleep disorders.

- **Identification:** The initial amount of articles was based on those found in the main study, which was 39.
- **Screening:** The 39 articles was screened, leaving 17 inside the cathegory of potential possibility of being implemented in a mobile solution.
- **Eligibility:** Of the 17 articles, 3 were found to be directly useable in the project.
- **Inclusion:** The total amount of three references came out of the sub-study.



C Appendix: Experimental Documents

This chapter includes all of the documentation used to conduct the clinical trials. The first section includes all of the documents presented to the subjects when introducing them to the project. The second chapter includes the Case Report File (CRF) used by the resaercher to document the process of each individual recording in the experiment.

C.1 Documents in Clinical Trials

Rigshospitalet Glostrup

Technical University of Denmark DTU

Skriftlig deltagerinformation

Et smartphone baseret screeningssystem til detektion af søvnapnø, version 3

15. februar 2016

Dansk Center for Søvnmedicin
Rigshospitalet Glostrup

Sagsnr. 49333

1 Patient Information

Du er blevet henvist til Dansk Center for Søvnmedicin for at blive undersøgt for obstruktiv søvnapnø. Deltagelse i forskningsprojektet indbefater, at der samtidig med den klassiske undersøgelse til diagnosticering af sønvapnø vil blive foretaget yderligere målinger. Det betyder, at deltagelse i forsøget vil forlænge din undersøgelse med maksimalt 20 minutter. Det er standard procedure, at undersøgelse for obstruktiv øvnapnø foregå ambulant. Dette betyder, at du får udleveret udstyr til undersøgelse på hospitalet, men at du skal sove med udstyret på, derhjemme.

Forsøget vil ikke blive brugt diagnostisk i forhold til forsøgspersonerne, men diagnosen vil som vanligt blive foretaget på baggrund af standard undersægelsen. Forsøgspersonerne har derfor ikken ogen direkte nytte af at deltage i forsøget udover, at en validering af systemet vil bidrage til videnskaben og dermed i fremtiden kunne forbedre diagnosticeringen af søvnapnø.

Forsøget er en afprøvning af et nyt smartphone baseret system til at afgøre, om patienter er i højrisikogruppen for at have søvnapnø. Formålet med dette er, at hvis metoden viser sig at være effektiv nok, skal patienterne inden henvisning til den fulde undersøgelse hae lavet denne forundersøgelse. Dette vil gøre, at færre patienter fremover skal have foretaget en fuld savnanalyse kaldet en polysomnografi eller en polygrafi, hvilket der er flere fordele ved. Den nye undersøgelse kan laves i hjemmet, således patienten slipper for at skulle på hospitalet. Samtidig er en fuld søvnanalyse en dyr unersøgelse, og man vil derved kunne opnå besparelser i sundhedsvæsnet.

Til forsøget vil der blive placeret en smartphone på brystkassen, og da undersægelsen er ambulant, vil du selv skulle betjene smartphonen. Det skal dog fremhæves, at der blot skal trykkes start og stop, og det derfor er lige til at gå til. En mikrofon vil blive placeret ved sin af næsen for at optage din vejrtræning og eventuel snorken undersøvnen.

Inden forsøget startes, skal du svare på et spørgeskema på telefonen. Spørgeskemaet er videnskabeligt udviklet, og det består af otte ja/nej spørgsmål, Tabel 1. Desuden vil en sensor der er indbygget i smartphonen, registrere bevægelser, hvilket kan bruges til at bestemme ventilationsmønster, graden af bevægelser, hjertefrekvens og positioner under optagelserne. Efter undersøgelsen er slut er der ligeledes nogle spørgsmål. Disse er for at afdække din oplevelse med undersøgelsen. Spørgeskemaet er på fire spørgsmål, der kan ses i Tabel 2.

Spøgsmål STOP-BANG
Køn
Alder
BMI
Snorker du højlydt?
Føler du dig ofte træt, udmattet eller søvnig i løbet af dagen?
Har nogen observeret dig stoppe med at trække vejret i søvne?
Har du højt blodtryk, eller bliver du behandlet for det?
Er dine Nakke omkreds over 40 cm?

Spørgsmål til efter undersøgelsen
Havde du nogen problemer med at udfører eksperimentet? - hvis ja, hvilke?
Påvirkede forsøgsopstillingen din søvn? - hvis ja, på hvilken måde?
Fandt du det ubehageligt at have udstyret på? - hvis ja, på hvilken måde?
Har du yderligere spørgsmål?

1.1 Udvælgelse af forsøgspersoner

Til forsøget er der blevet stillet en række eksklusions- og inklusionskriterier. For at være med i undersøgelsen skal du være henvist til Dansk Center for Søvnmedeicin på Rigshospitalet Glostrup til udredning for søvnapnø. Forsøgspersonerne er patienter, der er over 18 år og habile. Desuden må de ikke være gravide eller have en BMI over 40 $\frac{kg}{m^2}$. En forsøgsperson kan udelukkes fra videre deltagelse iforsøget af flere årsager: der opstår problemer under forsøget med monteringen af udstyret, eller den opsamlede data indeholder fejl. Desuren kan forsøgspersonen til enhver tid trække sin accept tilbage med hensyn til deltagelse i forsøget.

1.2 Hvis du ikke ønsker at deltage i forsøget

Hvis du ikke ønsker at deltage i forsøget, vil du få foretaget den almindelige undersøgelse, og du vil selvfølgelig få den samme behandling som alle andre.

1.3 Risiko og bivirkninger ved deltagelse

Der er ikke nogle risici ved at deltage i dette forsøg. Af bivirkninger kan det nævnes, at er kan være ubehag at ligge med en smartphone på brystkassen.

1.4 Videregivelse af patientinformationer

Ved at underskrive samtykkeerklæringen vedrørende patientinformation giver du din accept til videregivelse af nødvendige oplysninger om dine helbredforhold, øvrige rent private forhold og fortrolige oplysninger som et led i kvalitetskontrol og monitorering. Fra din patienthournal vil der blive videregivet data fra din søvnunderøgelse samt lægens endelige konklusion på undersøgelsen vedrørende øvnapnø hændelser og diagnosticering. Det skal understreges, at patientinformationer er strengt fortrolige og forkserne bag forsøget er underlagt tavshedspligt. Oplysningerne fra patientjournalen skal bruges tilat validere det testede system. Dette er til for at konstatere, om systemets konklusioner stemmer overens med den gældenen stendard for registrering af søvnapnø hændelser og diagnosticering af søvnapnø.

1.5 Økonomi

Forsøget er et inginiør speciale, hvor det er Danmarks Tekniske Universitet, der har taget initiativet til at lave forskningsprojektet. Udgifterne dækkes af Danmarks Tekniske Universitet og Rigshospitalet Glostrup i fælleskab. Forskerne har intet økonomisk udbytte af projektet.

1.6 Hvem får at vide, at jeg deltager i forsøget

Samtykket omfatter adgang til videregivelse og behandling af nødvendige oplysninger om forsøgspersonens helbredsforhold, øvrige rent private forhold og andre fortrolige opslysniger som led i relevante myndigheders lovpligtige kontrol med forsøget.
Alle personer involveret i forsøget har tavshedspligt og vil behandle dine personlige data strengt fortroligt. Der vil på intet tidspunkt blive udleveret oplysninger, som kan henføres til dig personligt. Hvis du vælger at trække dit informerede samtykke tilbage, vil ingen nye data blive indsamlet og registreret. Imidlertid tillader lovgivningen, at data indsamlet inden du trækker dit samtykke tilbage stadig indgår i forsøgets datamateriale. Oplysningerne vil blive registretet og opbevaret i 15 åt efter forsøget afslytning, og de bil blive anvent i en videnskabelig opgørelse.

1.7 Frivilig Deltagelse

Meningen med denne skriftlige information er, at ud i ro og mag skal kunne overveje situationen og drøfte den med dine nærmeste. Du har ret itl betænkningstid, ligesom vi vil anbefale, at du har en pårørende med til samtalen. Inden du beslutter, om du vil deltage i forsøget, vil vi bede dig om at læse folderen "Før du besluter dig". Folderen er udgivet af Videnskabsetisk Kometé og er skrevet in på de sidste sider i denne information.

Det er din egen frivillige beslutning, om du vil deltage i forsøget. Hvis du beslutter dig for at deltage, kræver dansk lov, at u bekræfter dette ved at skrive uder på en samtykkeerklæring. føler du, at der bliver lagt pres på dig, råder vi dig til at udskyde afgørelsen. Vælger du at deltage kan du i øvrigt når som helst og uden begrundelse tilbagekaled dit samtykke og din fuldmagt. Uanset om du siger ja, nej eller fortryder senere, vil vi give dig de bedst mulige vilkår og undersøgelser. Forsøget er godkendt af Videnskabsetisk Komité (sagsnr. 49333). Videnskabetisk Komités govedopgave er at beskytte forsøgspersoner, der deltager i biomedicinske forsøg og at sikre at man som forsøgsperson frit kan vælge, om man vil deltage.

Med venlig hilsen

Professor overlæge dr.med Poul Jennum
Dansk Center for Søvnmedicin, Glostrup Hospital
Nordre Ringvej 57,5.Opgang 5
2600 Glostrup.

Stud polyt Mathias Pinto Bonnesen
Institut for Elektroteknologi Danmarks Tekniske Universitet
2800 Kongens Lyngby
2600 Glostrup.

DEN NATIONALE VIDENSKABSETISKE KOMITÉ

The National Committee on Health Research Ethics

Revideret august 2014

Før du beslutter dig

Om at være forsøgsperson i sundhedsvidenskabelige forsøg

DEN NATIONALE VIDENSKABSETISKE KOMITÉ * HOLBERGSGADE 6 * 1057 KØBENHAVN K * TLF.: +45 72 26 93 70 *
WWW.DNVK.DK * DKETIK@DKETIK.DK

Hvis du overvejer at deltage i forsøg

For at få ny viden om sygdomme og blive bedre til at behandle dem, er det vigtigt at forske på mennesker. Kun gennem forskning kan man blive bedre til at forebygge, diagnosticere og ikke mindst at behandle sygdomme.

Du er muligvis blevet spurgt, om du vil deltage i et forsøg – eller du er interesseret i at få noget at vide om, hvad det vil sige at være forsøgsperson.

Hvis du er blevet spurgt, om du vil deltage – f.eks. af din egen læge eller af hospitalet, kan forsøget bl.a. dreje sig om en ny type medicin, supplerende undersøgelser eller en ny operationsmetode. Hvis du er rask og selv overvejer at henvende dig, evt. på baggrund af en annonce, kan forsøget f.eks. handle om kroppens funktioner, livsstil eller forebyggelse af sygdomme.

Nogle ønsker at deltage i et forsøg i forventning om at opnå en bedring af deres sygdom, andre ønsker at deltage for at hjælpe forskningen og dermed fremtidige patienter. Uanset hvad du ønsker, kan denne pjece oplyse dig om de videnskabsetiske komiteer, som skal godkende alle sundhedsvidenskabelige forsøg i Danmark, hvad det vil sige at være forsøgsperson i sundhedsvidenskabelige forsøg og om dine rettigheder som forsøgsperson.

Pjecen oplyser også om forskellige typer af forskning, og om den information, du har krav på at få, inden du tager stilling.

De videnskabsetiske komiteer

De videnskabsetiske komiteer består af både fagfolk (sundhedsvidenskabeligt uddannede) og lægfolk – med et flertal af lægfolk. Komiteerne skal varetage forsøgspersonernes interesser. For selvom det er vigtigt at sikre sundhedsvæsenet ny viden, kommer hensynet til forsøgspersonerne altid først. Komiteerne skal derfor godkende alle sundhedsvidenskabelige forsøg i Danmark.

Før et forsøg kan godkendes, vurderer den videnskabsetiske komité både risikoen for forsøgspersonerne og nytten ved forsøget. Hvis et forsøg på nogen måde er uforsvarligt at gennemføre, bliver det ikke godkendt. Komiteen kan også afvise et forsøg, hvis den mener, at informationen til dig ikke er god nok. Den videnskabsetiske komité sikrer desuden, at der er en erstatnings- eller godtgørelsesordning, hvis en forsøgsperson skulle lide skade på grund af forsøget.

Om at være forsøgsperson

At være forsøgsperson vil sige, at man frivilligt deltager i et sundhedsvidenskabeligt forsøg. Forsøget udføres, fordi forskere mener, at den nye behandling måske er bedre eller mindst lige så god som den hidtidige standardbehandling. Men de ved det ikke med sikkerhed, og det er derfor, man laver forsøget.

Lovgivningen fastsætter, hvordan og på hvilke betingelser et forsøg må gennemføres. Lovgivningen er til for at sikre forsøgspersonens rettigheder, som gælder, uanset om deltageren er patient eller en rask person. Hvis du er patient og beslutter ikke at deltage, har du naturligvis altid sikkerhed for, at du alligevel vil modtage den bedste, kendte behandling (standardbehandlingen).

En del forsøg sammenligner standardbehandlingen med en ny forsøgsbehandling, som forskerne mener, kan være bedre. Det gøres ved at fordele deltagerne tilfældigt i to grupper – én, der får standardbehandlingen og én, der får forsøgsbehandlingen. Fordelingen kaldes randomisering og foretages for, at de to behandlinger kan sammenlignes bedst muligt.

Nogle kan føle utryghed ved den tilfældige fordeling af behandlinger. Men det er vigtigt at understrege, at de, der får forsøgsbehandlingen, muligvis får en bedre behandling end de, der får standardbehandlingen. De, der får standardbehandlingen, har til gengæld sikkerhed for at få en velafprøvet behandling. Hvis det undervejs i forsøget viser sig, at forsøgsbehandlingen er dårligere end standardbehandlingen, bliver forsøget afbrudt og alle forsøgspersoner vil derefter modtage standardbehandlingen.

Nogle forsøg tager flere år, og når et forsøg er afsluttet, gør forskerne resultaterne op. Herefter kan de vurdere, om forsøgsbehandlingen har vist sig bedre end standardbehandlingen, eller de kan bruge den nye viden til at gøre fremtidige behandlinger bedre.

Forsøgspersoners rettigheder

- Din deltagelse i forsøget er helt frivillig, og inden du eventuelt underskriver samtykkeerklæringen, skal du have både skriftlig og mundtlig information om forsøget.
- Du kan til enhver tid mundtligt, skriftligt eller ved anden klar tilkendegivelse trække dit samtykke til at deltage tilbage og dermed træde ud af forsøget. Selvom du trækker dit samtykke tilbage, påvirker dette ikke din ret til nuværende eller fremtidig behandling eller andre af dine rettigheder.
- Du har ret til at tage et familiemedlem, en ven eller en bekendt med som bisidder til informationssamtalen.
- Du har ret til betænkningstid, før du giver tilsagn om at deltage.
- Oplysninger om dine helbredsforhold, som kommer frem i forbindelse med forsøget, er omfattet af tavshedspligt. Det samme gælder øvrige rent private forhold og andre fortrolige oplysninger om dig.
- Opbevaring af oplysninger om dig, herunder oplysninger i/fra dine blodprøver og væv, sker efter reglerne i lov om behandling af personoplysninger og sundhedsloven.
- Du har mulighed for at få aktindsigt i forsøgsprotokoller efter offentlighedslovens bestemmelser. Det vil sige, at du kan få adgang til at se alle papirer vedrørende din deltagelse i forsøget, bortset fra de dele, som indeholder forretningshemmeligheder eller fortrolige oplysninger om andre.
- Du har mulighed for at klage og få erstatning efter reglerne i lov om klage- og erstatningsadgang inden for sundhedsvæsenet (se afsnittet om klagemuligheder).

For børn, voksne inhabile, bevidsthedsslørede og lignende gælder der særlige regler.

Det er frivilligt at deltage

Det er altid helt frivilligt at deltage i et sundhedsvidenskabeligt forsøg. Lægen kan derfor ikke fortælle dig, om du skal deltage eller ej – det er din helt egen beslutning.

Før du tager stilling til, om du vil deltage i et forsøg, skal du have både mundtlig og skriftlig information om forsøget. Den mundtlige information gives af den eller de personer, der har ansvar for forsøget, og det er vigtigt, at du benytter lejligheden til at stille spørgsmål, hvis du er i tvivl. Du har ret til at have en bisidder (pårørende, ven eller bekendt) med til samtalen om forsøget, så I er to til at stille spørgsmål og tale om din eventuelle deltagelse bagefter. Du har også krav på betænkningstid, før du beslutter dig – som regel mindst 24 timer.

Vælger du at sige ja til at blive forsøgsperson, skal du skrive under på, at du vil deltage, og du skal skrive under på, at du har modtaget både mundtlig og skriftlig information om forsøget. Men selvom du har skrevet under på, at du vil deltage, kan du *altid* træde ud af forsøget igen. Hvis du træder ud af forsøget, vil forskerne ofte gerne vide hvorfor, af hensyn til resultaternes pålidelighed, men du er ikke forpligtet til at oplyse det. Hvis du deltager som patient og træder ud af forsøget, vil du stadig modtage den behandling, som du ville have fået, hvis du ikke havde deltaget i forsøget. Du mister altså ingen rettigheder som patient ved at deltage i et forsøg.

Forskellige typer af forsøg

Et forsøg kan f.eks. dreje sig om, at man afprøver et lægemiddel, noget medicinsk udstyr eller måske en ny behandlingsmetode, f.eks. et forsøg, hvor man opererer på en anden måde end hidtil.

Når man afprøver et lægemiddel på mennesker, er det ofte for at undersøge lægemidlets effekt og sikkerhed og for at opdage eventuelle bivirkninger.

Forsøg med medicinsk udstyr drejer sig ofte om udstyrets sikkerhed og ydeevne. Medicinsk udstyr er udstyr, der bruges til at undersøge, behandle eller lindre sygdom (f.eks. sprøjter, operationsudstyr, plastre, pacemakere eller hofteproteser).

Afprøvning af lægemidler og medicinsk udstyr kræver en godkendelse fra både Sundhedsstyrelsen og en videnskabsetisk komité. Forsøg, der ikke involverer lægemidler skal udelukkende godkendes af det videnskabsetiske komitésystem.

Du skal have skriftlig information om

- At du anmodes om at deltage i et sundhedsvidenskabeligt forsøg, og at du modtager ikke-almindelig behandling.
- Formålet med forsøget, og hvordan det udføres.
- (I tilfælde af lægemiddelafprøvninger:) Hvilke godkendte og ikke godkendte lægemidler, der anvendes.
- Eventuel tilfældig fordeling mellem flere forskellige behandlingsformer, hvoraf én kan være behandling med et uvirksomt præparat (placebo), hvis der ikke findes en standardbehandling.
- Risici, bivirkninger og ulemper ved at deltage i forsøget.
- Nytten af forsøget.
- Hvis der i forbindelse med forsøget udtages biologisk materiale (f.eks. blod eller væv), der opbevares.

- (Hvis du deltager som patient:) Hvad der er almindelig behandling, om der er andre mulige behandlingsmetoder, og hvilken behandling du vil få, hvis du vælger at sige nej til at deltage i forsøget. Du skal også have oplyst, hvis der anvendes oplysninger fra din patientjournal.
- At oplysninger fra din patientjournal, rent private forhold og andre fortrolige oplysninger kan videregives til og behandles af personer, som skal foretage en lovpligtig kvalitetskontrol af forsøget.
- Under hvilke omstændigheder du kan udelukkes fra forsøget.
- Under hvilke omstændigheder forsøget kan blive afbrudt.
- Navn, adresse og telefonnummer på forsøgets kontaktperson.
- Eventuel økonomisk støtte, som forskeren modtager fra private virksomheder og fonde.
- Eventuelt vederlag for deltagelse og oplysning om beskatning heraf.

Den skriftlige information skal udleveres i god tid, så vidt muligt 24 timer inden du tager stilling til, om du vil deltage.

Krav til forskerne

Der stilles uddannelsesmæssige krav til den ansvarlige for forsøget afhængigt af forsøgets indhold. Den ansvarlige skal have ret til at udøve forskning – det kan f.eks. være via en ansættelse som forsker og vedkommende skal også have erfaring med at behandle patienter fra f.eks. en hospitalsafdeling.

Hvis der er tale om forsøg med lægemidler eller medicinsk udstyr, skal den ansvarlige have en uddannelse som læge eller (hvor det er relevant) tandlæge. Hvis deltagerne under forsøget får alvorlige bivirkninger eller der sker alvorlige hændelser, skal den forsøgsansvarlige omgående underrette den videnskabsetiske komité. Hvis der er tale om forsøg med lægemidler, skal Sundhedsstyrelsen også underrettes.

Vederlag, skattepligt mv.

I visse situationer betales der vederlag til forsøgspersoner. Der kan f.eks. være tale om tabt arbejdsfortjeneste, transportgodtgørelse, ulempegodtgørelse eller andet. Det vil fremgå af deltagerinformationen, om der gives vederlag i forbindelse med deltagelse i forsøget, og om det er skattepligtigt.

Klage- og erstatningsmuligheder

Hvis du som forsøgsperson bliver udsat for en utilfredsstillende behandling eller kommer til skade, er der forskellige klage- og erstatningsmuligheder.

Ved godkendelse af forsøget sikrer den videnskabsetiske komité, at der er en forsikrings- og erstatningsordning for forsøgspersonerne.

Ønsker du at søge erstatning for en skade i forbindelse med et forsøg inden for sundhedsvæsenet, skal du anmelde skaden til:

Patienterstatningen
Hjemmeside: www.patienterstatningen.dk
E-mail: pebl@patienterstatningen.dk Telefon: 33 12 43 43

Ordningen dækker også lægemiddelskader. Raske forsøgspersoner er også omfattet af reglerne om arbejdsskadesikring.

Er forsøget uden for det område, som Patienterstatningen dækker, skal der være tegnet en særskilt forsikring. Sådanne erhvervsansvarsforsikringer dækker normalt efter almindelige erstatningsretlige regler. Du kan få yderligere information af den ansvarlige for forsøget. I tvivlstilfælde kan du henvende dig til sekretariatet i den videnskabsetiske komité, der dækker din region.

Du har mulighed for at klage over den sundhedsfaglige behandling i forsøget. Du kan sende klagen til:

Patientombuddet
Hjemmeside: www.patientombuddet.dk
E-mail: pob@patientombuddet.dk
Telefon: 72 28 66 00

Du kan evt. få hjælp hertil fra regionernes patientvejledere. Se nærmere på www.sundhed. dk.

Hvis du vil vide mere

Vi håber, at du har fået et tilstrækkeligt indblik i, hvad det vil sige at deltage i et forsøg og føler dig rustet til at tage beslutningen om din eventuelle deltagelse. Hvis du vil vide mere om det at være forsøgsperson, er du meget velkommen til

at kontakte sekretariatet i en af de regionale videnskabsetiske komiteer. Du kan også finde flere oplysninger og spørgsmål/svar på www.dnvk.dk.

De Videnskabsetiske Komiteer for Region Hovedstaden (6 komiteer) Kongens Vænge 2
3400 Hillerød
Tlf. 38 66 63 95
E-mail: vek@regionh.dk
Hjemmeside: www.regionh.dk/vek

Den Videnskabsetiske Komité for Region Sjælland Alléen 15
4180 Sorø
Tlf. 24 52 59 52 / 57 87 52 44
E-mail: RH-komite@regionsjaelland.dk Hjemmeside: www.regionsjaelland.dk/videnskabsetisk-komite

De Videnskabsetiske Komiteer for Region Syddanmark (2 komiteer) Regionshuset
Damhaven 12
7100 Vejle
Tlf. 20 59 89 30 / 29 20 22 51 / 29 20 22 52 /
29 20 12 03
E-mail: komite@rsyd.dk
Hjemmeside: www.regionsyddanmark.dk/komite
De Videnskabsetiske Komiteer for Region
Midtjylland (2 komitéer)
Regionssekretariatet, Juridisk kontor
Skottenborg 26
8800 Viborg
Tlf. 78 41 01 81 / 78 41 01 82 / 78 41 01 83
E-mail: komite@rm.dk
Hjemmeside: www.komite.rm.dk

Den Videnskabsetiske Komité for Region Nordjylland
Regionssekretariatet
Niels Bohrs Vej 30
9220 Aalborg Ø
Tlf. 97 64 84 40
E-mail: vek@rn.dk
Hjemmeside: www.vek.rn.dk

Den Nationale Videnskabsetiske Komité
Holbergsgade 6
1057 København K
Tlf.: +45 72 26 93 70
E-mail: dketik@dketik.dk
Hjemmeside: www.dnvk.dK

August 2014

Samtykkeerklæringer

Der anvendes til forsøget to samtykkeerklæringer. VEK's standard samtykkeerklæring S1 samt samtykkeerklæringen til videregivelse af patientdata. Samtykkeerklæringen til videregivelse af patientdata er vedhæftet til orientering for VEK.

Samtykeerklæring til videregivelse af patientdata

Vedrørende deltagelse i forsøget: Bærbar device til analyse og fortolkning af sønvapnø

Jeg giver fuldmagt til, at der kan videregives projektrelevante oplysninger i min journal til den forsøgsansvarlige (jf. Sundhedsloven, lov nr. 546 af 24 juni 2005). Denne fuldmagt kan til enhver tid tilbagekaldes.

Alle oplysninger bliver behandlet fortroligt.

Udlånt Telefon:

Patientnavn:

Patient underskrift:

Dato:

DET VIDENSKABSETISKE KOMITÉSYSTEM

(S1)

Informeret samtykke til deltagelse i et sundhedsvidenskabeligt forskningsprojekt.

Forskningsprojektets titel: Bærbar device til analyse og fortolkning af sønvapnø

Erklæring fra forsøgspersonen:

Jeg har fået skriftlig og mundtlig information og jeg ved nok om formål, metode, fordele og ulemper til at sige ja til at deltage.

Jeg ved, at det er frivilligt at deltage, og at jeg altid kan trække mit samtykke tilbage uden at miste mine nuværende eller fremtidige rettigheder til behandling.

Jeg giver samtykke til, at deltage i forskningsprojektet, og har fået en kopi af dette samtykkeark samt en kopi af den skriftlige information om projektet til eget brug.

Forsøgspersonens navn: _____

Dato: _____ Underskrift: _____

Ønsker du at blive informeret om forskningsprojektets resultat samt eventuelle konsekvenser for dig?:

Ja _____ (sæt x) Nej _____ (sæt x)

Erklæring fra den, der afgiver information:

Jeg erklærer, at forsøgspersonen har modtaget mundtlig og skriftlig information om forsøget.

Efter min overbevisning er der givet tilstrækkelig information til, at der kan træffes beslutning om deltagelse i forsøget.

Navnet på den, der har afgivet information: Mathias P. Bonnesen

Dato: _____ Underskrift: _____

Projektidentifikation: (Fx komiteens Projekt-ID, EudraCT nr., versions nr./dato eller lign.)

Sagsnr. 49333

INFORMATION

Denne brugervejledning forklarer hvordan du punkt-for-punkt, fastsætter måleudstyret og igangsætter dataopsamlingen, i forbindelse med projektet "Mobile Apnea Screening".

Dataopsamlingen vil være meget værdifuld for videreudviklingen af det mobile screenings system. Du skal have mange tak for din deltagelse i projektet.

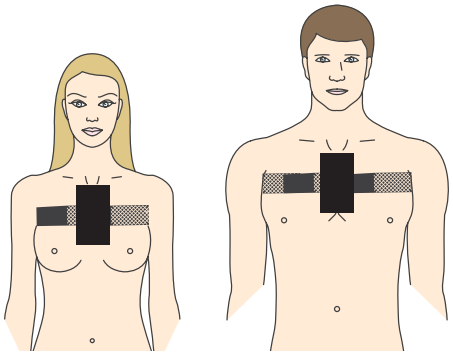
Med venlig hilsen
Mathias P. Bonnesen
Specialestuderende

I samarbejde med
Danmarks Tekniske Universitet
og Rigshospitalet - Glostrup

MOBILE APNEA SCREENING

BRUGERVEJLEDNING

1) PLACERING AF REM OG HOLDER

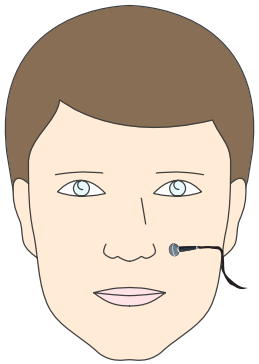


A Placér remmen rundt om brystet og lige under armhulerne. Det gennemsigtige plastik skal vende udad og åbningen til telefonen skal vende opad.

B Spænd remmen fast ved hjælp af velcroen for enden af remmen. Husk, at du skal sove med den hele natten, så spænd ikke remmen for stramt. Dog er det vigtigt, at remmen forbliver på sin korrekte placering hele natten.

C Justér placeringen af hylsteret til mobiltelefonen således, at den øverste del af hylsteret er placeret ved kravebenet.

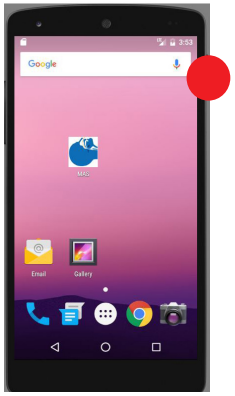
2) PLACERING AF MICROFONEN



Placér mikrofonen på kinden, så den peger vandret mod næsen, uden at være under næseborene, som anvist på illustrationen til venstre.

Når du har placeret mikrofonen korrekt, kan du klæbe den fast med et stykke tape/plaster.

3) IGANGSÆTTELSE AF DATAOPSAMLING



1 Du tænder for telefonen ved at holde knappen på højre side af telefonen nede i 4 sekunder (til skærmen tænder). Anvist med en rød prik i illustrationen til venstre.

2 Når telefonen er tændt, låser du telefonen op ved at stryge fingeren opad på skærmen.

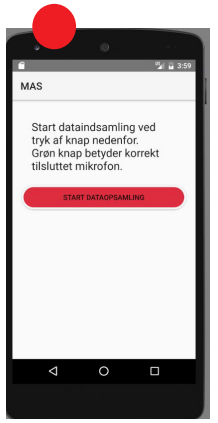
3 Åben applikationen ved at trykke på symbolet 

4 Vælg køn ved at trykke på "svar".

5 Indtast alder og tryk på det grønne flueben nederst til højre (se figur til venstre)

6 Indtast din vægt i kg og højde i cm, ved at trykke på "KLIK HER" og dernæst "OK". Se billedet til højre.

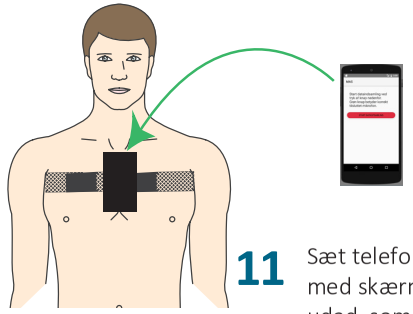
7 De resterende spørgsmål besvarer du som i trin (4) og trykker derefter på fluebenet helt oppe i højre hjørne.



8 Tilslut mikrofonen i toppen af telefonen (se rød prik i figuren til venstre). Når mikrofonen er tilsluttet, bliver den røde bjælke "Start dataopsamling" grøn.

9 Tryk på den grønne bjælke "Start dataopsamling" for at starte dataopsamlingen.

10 Sluk skærmen på samme knap som du tændte (trin 1), men tryk nu blot kortvarigt.



11 Sæt telefonen i hylsteret med skærmen vendt udad, som anvist i illustrationen.

NU KAN DU LIGGE DIG TIL AT SOVE.

4) NÅR DU VÅGNER

Tag telefonen ud af hylsteret og tænd skærmen ved at trykke på samme knap, som i trin (11). Afslut dataopsamlingen ved at trykke på knappen "STOP". Du kan nu slukke telefonen på samme måde som du tændte den, som i trin (1).

HUSK at aflevere udstyret tilbage til klinikken.

C.2 Experimental Protocol

Rigshospitalet Glostrup Technical University of Denmark DTU

Mobile Apnea Screening (MAS)
- Experimental Protocol

February 15, 2016, Version 4

Mathias P. Bonnesen, Helge B.D Sørensen and Poul Jørgen Jennum

Project summary

Sleep apnea is a sleep disorder that an estimated 2-4 % of the general population suffers from. It comes in three types; obstructive sleep apnea (OSA), Central Sleep Apnea (CSA) or mixed, where 90 % are diagnosed with OSA [[1]]. The diagnosis is done using a Polysomnograph (PSG) to measure physiological parameters during a full nights sleep, which is a costly procedure. Furthermore, an estimated 50 % of subjects referred to Glostrup Rigshospital is found to be healthy.

Therefore, this study propose a new cheaper method which can bring down the percentage of healthy subjects referred to PSG studies. It will contain only two data modalities; Audio and accelerometer measurements, to screen for OSA and determine the severity of the disease. The microphone is placed on the cheek horizontally pointing towards the nostrils placed on line with them vertically, fixating it with sticky tape. The accelerometer measurements are acquired through a smartphone tri-acis accelerometer placed on sternum, strapped on with a modified smartphone armband. Data acquisition from the sensors is done through a custom made android application, which also gathers demographic information by making the user answer a questionnaire.

The experiment is conducted on subjects referred to PSG tests on Glostrup Righshospital, where the data acquisition is conducted parallel to the PSG study. The PSG will be manually scored by medical professionals resulting in the severity index, Apnea-Hypapnea Index (AHI) which will be the results on which the performance of the proposed solution will be scored against. The statistical measures of performance will be *sensitivity*, *specificity* and *accuracy*.

The study is expected to result in an scientific publication, and the data acquired will be part of a database at Glostrup Rigshospital for use in other projects.

Indhold

1 Rationale & background information

There are in general three types of sleep apnea diagnoses; obstructive sleep apnea, central sleep apnea (CSA) or mixed. OSA is defined as a complete or partial obstruction of the pharynx, CSA happens when the brain fails at activating the muscles in the breast accountable of breathing and mixed is a mix between the two.

OSA is the most prevalent of the two with an estimated 2-4 % of the adult population globally suffering from it, with a variance across ethnic groups[1]. The problem arises due to what is called *apneic moments* which is defined as partial or complete closing of the airway. This causes a decrease in oxygen inflow eventually triggering an arousal which makes the person suffering from OSA, wake up. Frequency of an arousal depends on the severity of the disease, and varies from 5 and hour to more than 50 and can therefore highly disrupt the sleeping pattern. Depending on the severity of the disease, it is known result in a wide range of clinical manifestations including daytime sleepiness and automobile accidents, to memory loss, cardiovascular diseases,

depression and more[6].

Currently the polysomnography (PSG) is the gold standard when diagnosing OSA. However devices capable of measuring parameters associated with Sleep apnea have been developed, and are typically placed into one of 4 groups (where PSG is group 1)[3]. An increase in the complexity of the screening process is related to a lower gruop number (see section **??**). The problem lies in that an estimated 50 % of patients undergoing PSG to diagnose OSA proves to be healthy. Since the PSG is relatively expensive method, this reveals opportunities of saving a lot of money by reducing the healthy individuals undergoing PSG. Previous studies regarding this matter have been conducted, proposing a broad range of solutions. This study focuses on developing a simple mobile solution with no more than three sensors, taking into account published state of the art articles[5] [6] [7] [8] [9] [10] [11]. They all have in common that they use different variations of three different sensors; microphone, accelerometer from smartphone and an oximeter. The physiological signals they measure and the signal processing techniques used to process the signals are very different, and not all have been validated with the PSG.

2 Study goals and objectives

The scope of this research study is to develop a novel OSA screening method capable of grouping patients into two categories; *Healthy* and *non-healthy*, and the latter into three subcategories; *mild*, *moderate* and *severe OSA*. The solution aims to be simple and easy to perform, so that it can be conducted pre-PSG to investigate if a PSG is necessary at all. The goal is to reduce the number of healthy individuals undergoing PSG to under 5 %, saving patients unnecessary inconvenience and the national health system a lot of money.

3 Study Design

The experimental design is non-randomized, case-series where the population for this study consists of patients who have been referred to "Dansk Center for Søvnmedicin"at Rigshospitalet Glostrup for an investigation if they suffer from OSA. Empirically it is estimated that 50 % of the referred patients are diagnosed with OSA and 50 % do not have OSA.

The aim of the study is to reduce the number of medical examinations on patients who end up being diagnosed negatively regarding OSA from 50 % to an estimate of 5 %. The open source tool www.openepi.com has been used to perform a power calculation estimating the sample size needed to be performed to obtain a statistic well-founded study. The inputs for the power estimation can be found in Table 1.1. The Kelsey method estimates that 42 patients must be included in the study where the Fleiss method estimates a sample size of 38 to be enough at a 95 % significance level and a power of 90%. Due to these estimates the aim is to include 42 tests in this study.

3.1 Inclusion standards

- Patients referred to the sleep clinic at Glostrup Rigshospital for diagnosing of OSA via PSG.
- Patients between the age of 18 and 70 years old.
- Patients who choose to participate in this study.

Tabel 1: 1

Parameter	Value
Two-sided significance level (1-alpha) %	95
Power (1-beta) %	90
Ratio of sample size, Unexposed/Exposed	1
Unexposed with Outcome %	50
Exposed with Outcome %	4.8
Odds ratio	0.05
Risk/Prevalence Ratio	0.1
Risk/Prevalence Difference	-4.5

3.2 Exclusion standards

- Patients who are pregnant.
- Patients with an BMI above 40 kg/m^2
- Patients who by the medical staff is found to not be able to cooperate sufficiently for the data gathering to be conducted properly.

4 Material and Methods

4.1 Devices

The device used for the data gathering process and as accelerometer is a LG Nexus 5. The microphone used for sound acquisition is a Røde SmartLav+ microphone and the fixation gear for chest fixation is a PURO armband modified by extending the strap with a cloth band having Velcro on one end. The specifications can be seen below

Tabel 2: LG Nexus 5 specifications.

Dimensions (L-W-D in mm)	Battery time Talking (3G)	Acc. fs awake (hz)	Acc. fs in Sleep)
137.9-69.2-8.6	17 hours	Normal: 15 Game: 50.05 Fastest: 196	Normal: 4.96 Game: 50.12 Fastest: 198

Tabel 3: Rode SmartLav+ specifications.

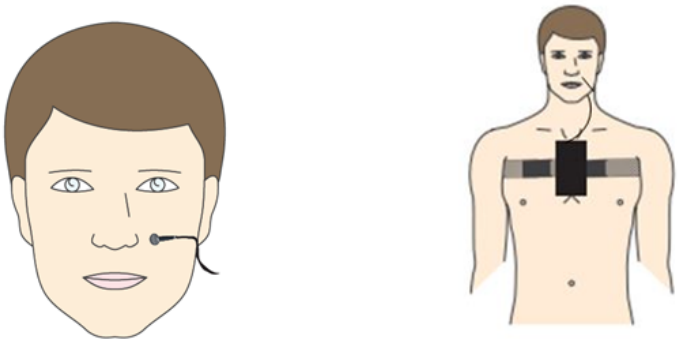
Frequency Response	dB-scale	S/N Ratio	Weight
20Hz - 20kHz	-35dB	115 dB	6g

Tabel 4: PURO armband specifications

Material	Max screen size	Headphone ports	Weight
Neoprhen	5 inches	One in each corner	60g

4.2 Experimental setup

The experimental setup includes a smartphone, a microphone, and a custom made smartphone holder that can be strapped around the chest. The microphone was placed on the cheek using sticky tape, pointing horizontally towards the nose with a vertical position on line with the nostrils almost touching the nose. The smartphone is placed on the location of the upper sternum with the screen pointing out and the microphone input pointing up. See the experimental setup below.



Figur 1: The audio is gathered from an extern microphone placed on the cheek pointing against the nose, without touching it. The smartphone is strapped firmly to the chest with the velkro-straps so that the top of the smartphone (where the microphone is plugged in) is around supra-sternum.

The above experimental setup is set up just before the patient goes to sleep, and can be taken off as soon as they wake up. The data acquisition is handled through an android application specifically developed for the study. Furthermore, the android application is also used to gather demographic data about the patient by making them answer a questionnaire.

4.3 Questionnaire

This is used to gather demographic information regarding the patient, and is a modified version of the STOP-BANG questionnaire often used when diagnosing OSA. See the questions below

- What is your gender?
- How old are you?
- What is your BMI?
- Do you snore loudly?
- Do you often feel tired, exhausted or sleepy during the day?
- Have anyone ever observed you stopped breathing during your sleep?
- Do you have high blood pressure or are you currently being treated for it?
- Does you neck-circumference exceeds 40 cm? (shirt-size)

The modifications lies in the three first questions, which has been changed from being binary yes/no questions. This is to be able to provide statistical knowledge of the study participants when the study is finished.

4.4 Audio acquisition

The audio captured from the microphone on the cheek is meant to provide information regarding airflow and snoring sounds. The airflow will be used, together with the accelerometer readings to determine breaths taken. An android application activity is used to access, measure and save the audio recording into a file.

4.5 Accelerometer

The accelerometer readings will have three different uses; create an actiography of the night, classify the readings into different sleep positions and find breaths taken. The actiography will provide information about arousals occurring during the night. An android application activity is used to access, measure and save the accelerometer readings into three files; one for each accelerometer axis (x,y and z).

4.6 Data Evaluation and Analysis

The acceleration data acquired by the accelerometer will be used to analyze the movement of the subject during the night, looking for patterns describing events such as arousals, body position change etc.

The audio acquired by the microphone will be used to analyze on both airflow and snoring sound of the subject during the night.

The questionnaire is used to implement the risk factors into the probability of the subject having OSA. Each question answered with a *yes*, gives one point and a *no* gives zero. A higher point total from all questions indicates a higher probability of having OSA.

4.7 Study evaluation and Classification

To validate the study, the results obtained with the proposed method will be compared to gold standard measurements conducted with a PSG. This will be done by using the statistical methods *sensitivity*, *accuracy* and *specificity*.

5 Safety Considerations

There are no safety hazards associated with the experiment. The questions in the questionnaire are basic, and not of any intimate nature. Since the patients undergo a diagnostic PSG while the data for the study is acquired, the medical examination will not be affected participating in the project. The experimentalt is designed in a way that does not influence the PSG results.

6 Expected Outcomes of the Study

The data acquired through the experiment will be part of a clinical database of sleep studies, to be used in future projects. The project itself, will use this data to investigate the validity of physiological signals measured with the proposed method, in relation to OSA. Success of the project would result in a large reduction of patients being referred to the sleep clinic for a PSG diagnostic, saving the clinic a lot of time and money.

The data acquired through this study is planned to be part of a scientific paper.

7 Dissemination of Results and Publication Policy

All participants in the study will receive a random label, making all of the data anonymous. The audio signal recorded will be downsampled to 4 kHz, making speech unrecognizable. The researchers for the article will be Mathias P. Bonnesen, Helge B.D. Sørensen and Poul Jennum.

8 Ethics

8.1 Protection of the patient

The persons performing this study will ensure that the study is in agreement with the Helsinki Accords or lows and rules in the country where the study is performed according to which part that gives the best protection of the patient. The protocol has been created, and studies will be done according to ICH harmonization guidelines for good clinical practice. The protocol will be approved by the local Committee on Health Research Ethics in accordance with the national instructions and laws in the participating centers before any studies are initiated. Furthermore the study will comply with the law regarding personal data.

8.2 Identification of the participants

A sequentially identification number will automatic be given to each patient who are registered in the study. This identification number will be entered in the application and all acquired data will be labeled with the number.

8.3 Data Protection Agency

Patient data will be added to a closed database at the department where the study is being performed and stored in accordance to the current regulations of the hospital. The study will be approved in accordance with the Danish Data Protection Agency (Datatilsynet) in accordance to the central registration by Region Hovedstaden.

8.4 Patient Approval

Specific patient information notes have been produced for this study. All patients will be informed about the aim of the study, possible risks, adverse effects, and inconveniences. The patients will be informed about the strict confidentially status regarding their patient data. It will be emphasised that participating in the study is strictly voluntarily, and that the patients at all times can refrain from any further participation in the study, and it will not influence on the 8 Sagsnr. 49333 Forsøgsprotokol patients' further treatment. Documented approval of participating in the study must be collected for all patients, before they are registered as participants. This must be done in agreement with ICH harmonization guidelines for good clinical practice. This entails: "den skriftlige informerede samtykkeform skal være underskrevet og personligt dateret af patienten eller af patientens juridisk acceptable repræsentant".

The patient information will be done according to the standard procedure at the department where the study will be performed. Since the examination is ambulant the patient information letter will be send to the patient but there is not a preliminary conversation regarding the examination. When the patient arrives at the department to be prepared for the examination the staff will also inform the patient about this scientific study. The patient will be asked if

he/she would like to participate in the study. The patient will be informed about the study in a quiet room at the medical department "Dansk Center for Søvnmedicin" study by the medical staff who writes the medical chart and informs about the coming examination. Both through the handed out written material and the oral information from the doctor the patient will be informed that the patient has the right to a companion when deciding. The study is considered by the researchers to be a relatively simple study with no risks and a small degree of inconvenience for the patient. After the introduction by the medical staff the patient is welcomed to ask questions, and when the patient feels fully informed the patient must decide if he/she wants to participate. If the patient is not capable of making a decision the patient will be excluded from the study. This procedure has been chosen since the standard procedure of the current examination of sleep apnoea does not include a preliminary information session why a longer deliberation time is impossible to obtain without introducing further inconveniences to the patient.

8.5 Pros and cons for the patients

There is no direct advantage for the patient participating in this study since the gold standard analysis will be performed simultaneously and used for the diagnosis. Since the study is performed simultaneously as the gold standard it will take a bit longer to prepare the analysis and the experimental setup might be a bit more unpleasant since more devices are added.

8.6 Organisation and publication of data

The study is a part of an engineering master's thesis in medicine and technology at Technical University of Denmark (DTU) and University of Copenhagen. The master's thesis is performed in corporation with "Dansk Center for Søvnmedicin"at Rigshospitalet Glostrup which is where all the examinations will be performed. The results will no matter the outcome be published in the final master's thesis and probably published in one or more scientific journals according to the Vancouver protocol.

9 Budget

The initiative to the project has been taken by DTU and it is done in cooperation with Rigshospitalet Glostrup. The expenses will be covered by the involved parts why there will be no funding by private corporations. The patients will not receive any economic compensation.

10 References

Litteratur

[1] Berger M. et al.,*Avoiding the supine position during sleep lowers 24 h blood pressure in obstructive sleep apnea (OSA) patients*, J Hum Hypertens 1997;11:657-64.

[2] Finkel J. K. et al.,*Prevalence of undiagnosed obstructive sleep apnea among adult surgical patients in an acedemic medical center*, Sleep Medicine 2009;10:753-758.

[3] Bucklin C. L. et al.,*An Inexpensive Accelerometer-Based Sleep-Apnea Screening Technique*, Naecon 2010:396-399.

[4] Al-Mardini M. et al.,*Classifying obstructive sleep apnea using smartphones*, Journal of Biomedical Informatics 2014;52:251-259.

[5] Nakano H et al.,*Monitoring Sound To Quantify Snoring and Sleep Apnea Severity Using a Smartphone: Proof of Concept*, Journal of Clinical Cleep Medicine 2014;10:73-78.

[6] Al-Mardini M. et al.,*Classifying obstructive sleep apnea using smartphones*, Journal of Biomedical Informatics 2014;52:251-259.

[7] Behar J. et al.,*SleepAp: An Automated Obstructive Sleep Apnoea Screening Application for Smartphones*, IEEE Journal of Biomedical and Health Informatics 2015;19:325-331.

[8] Oliver N. et Flores-Mangas F.,*HealthGear: Automatic Sleep Apnea Detection and Monitoring with a Mobile Phone*, Journal of Communications 2007;2:1-9.

[9] Alwassim S. et al.,*Sleep Apnea Monitoring Using Mobile Phones*, IEEE 14th International Conference on e-Health Networking, Applications and Services 2012.

[10] Morillo D. S et al.,*An Accelerometer-Based Device for Sleep Apnea Screening*, IEEE transactions on information technlogy in biomedicine 2010;14:491-499.

[11] Shi C. et al.,*Apnea MedAssist II: A smart phone based system for sleep apnea assessment*, Bioinformatics and Biomedicine, IEEE International Conference 2013:572-577.

C.3 Case Report File

This is the full Case Report File (CRF) used when performing the clinical trials, in order to keep a good experimental practice. The Names of all the participants have been anonymized as NA (Not Available). The cells covered in red are the subjects that got received the equipment with them home, but did not perform the experiment.

CASE REPORT FILE (CRF), - MAS CLINICAL EXPERIMENTS										
PatID	Telefone Info	Name	Date afl telefon	Scanned/ arch documents	Scanned BOTH	Project feedback	Recieved and Data loaded	Clean	CRM Data / Analysis	AHI
Pat 1	Telefon 1	NA	16-Nov-16	Yes	Yes	Yes	Yes	Yes	Yes	7.2
Pat 2	Telefon 2	NA	17-Nov-16	Yes	Yes	Yes	Yes	Yes	Yes	46.1
Pat 3	Telefon 3	NA	17-Nov-16	Yes	Yes	Yes	Yes	Yes	Yes	36.1
Pat 4	Telefon 1	NA	21-Nov-16	Yes	Yes	Yes	Yes	Yes	Yes	23.2
Pat 5	Telefon 3	NA	22-Nov-16	Yes	Yes	Yes	Yes	Yes	Yes	14.6
Pat 6	Telefon 2	NA	22-Nov-16	Yes	Yes	Yes	Yes	Yes	Yes	51.2
Pat 7	Telefon 3	NA	23-Nov-16	Yes			Yes	Yes	Yes, no edf	28.7
Pat 8	Telefon 1	NA	23-Nov-16	Forgot to sign permission			Yes	Yes	Yes	8
Pat 9	Telefon 2	NA	24-Nov-16	Forgot to sign permission			Yes	Yes	Yes	13.4
Pat 10	Telefon 3	NA	24-Nov-16	Yes	Yes	Yes	Yes	Yes	Yes	6.5
Pat 11	Telefon 1	NA	25-Nov-16	Yes			Recieved, NO DATA	Yes	NO	
Pat 12	Telefon 3	NA	30-Nov-16	Yes	Yes	No	Yes	Yes	Yes	82.6
Pat 13	Telefon 2	NA	30-Nov-16	Yes	Yes	Yes	Yes	Yes	Yes	21
Pat 14	Telefon 1	NA	30-Nov-16	Yes	Yes	Yes	Yes	Yes	Yes	4
Pat_15	Telefon 1	NA	1-Dec-16	Yes	Yes	Yes	Yes	Yes	Yes	13.1
Pat_16	Telefon 2	NA	1-Dec-16	Forgot to sign permission			Yes	Yes	Yes	27.9
Pat_17	Telefon 3	NA	1-Dec-16	Yes			Recieved, NO DATA	Yes		
Pat_18	Telefon 3	NA	2-Dec-16	Yes	Yes	Yes	Yes	Yes	Yes	5
Pat_19	Telefon 1	NA	5-Dec-16	Yes	Yes	Yes	Yes	Yes	Yes	3.1

Pat_20	Telefon 2	NA	5-Dec-16	Yes			Recieved, NO DATA	Yes		
Pat_21	Telefon 3	NA	5-Dec-16	Yes			Recieved, NO DATA	Yes		
Pat_22	Telefon 1	NA	6-Dec-16	Yes	Yes	Yes	Yes	Yes	Yes	37.1
Pat_23	Telefon 2	NA	6-Dec-16	Yes	Yes	Yes	Yes	Yes	Yes	32
Pat_24	Telefon 3	NA	6-Dec-16	Yes	Yes	Yes,mail	Yes	Yes	Yes	25.8
Pat_25	Telefon 1	NA	7-Dec-16	Yes	Yes	Yes	Yes	Yes	Yes	4
Pat_26	Telefon 2	NA	7-Dec-16	Yes	Yes	Yes	Yes	Yes	Yes	40
Pat_27	Telefon 1	NA	8-Dec-16	Yes	Yes	Yes	Recieved, NO DATA			
Pat_28	Telefon 2	NA	8-Dec-16	Yes	Yes	Yes	Recieved, NO DATA			
Pat_29	Telefon 2	NA	9-Dec-16	Yes	Yes	Yes	Yes	Yes	Yes	59.8
Pat_30	Telefon 1	NA	9-Dec-16	Yes	Yes	Yes	Yes	Yes	Yes	2.6
Pat_31	Telefon 1	NA	13-Dec-16	Yes	Yes	Yes	Yes	Yes	Yes	42.4
Pat_32	Telefon 2	NA	13-Dec-16	Yes	Yes	Yes	Yes	Yes	Yes	16.3
Pat_33	Telefon 1	NA	15-Dec-16	Yes	Yes	Yes	Yes	Yes	Yes	5.3

D Appendix: Screening Algorithm

This chapter includes different plots and figures associated to the design of the algorithm. The second section contains the phase- and amplitude response of each filter used in the preprocessing of the project. The third section contains figures and tables related to the classification of the dataset. Finally, the fourth and last section contains an image of how GUI of the created Visualization tool.

D.1 Preliminary Tests

This section contains the plots from the preliminary tests, which was used to determine that the system was ready for clinical trials.

General signal acquisition

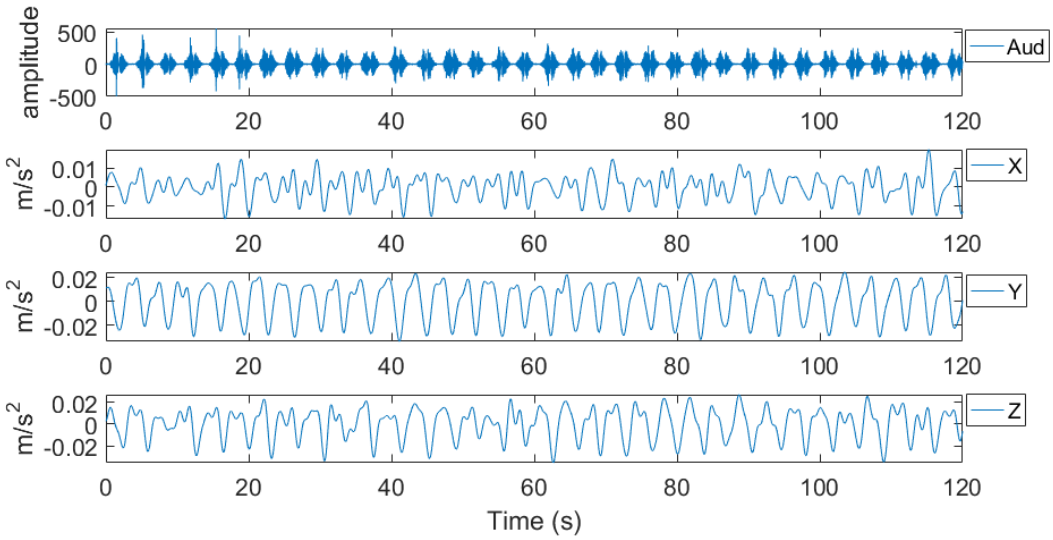


Fig. D.1: Segment of all signals captured from a sleeping family member in the preliminary test. This was to show how the signals looked when the subject was breathing without snoring.

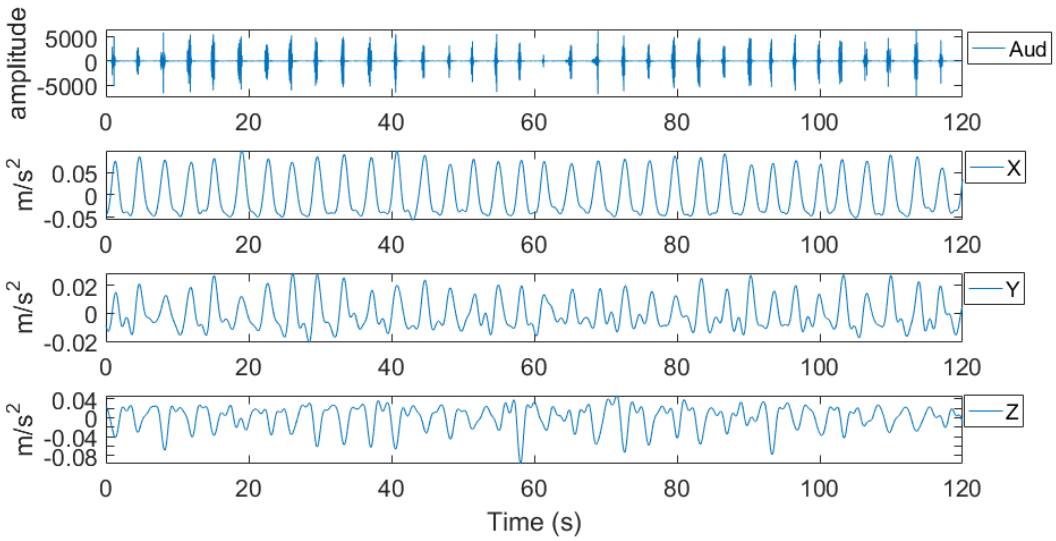


Fig. D.2: Segment of all signals captured from a sleeping family member in the preliminary test. This was to show how the signals looked when the subject was snoring while sleeping.

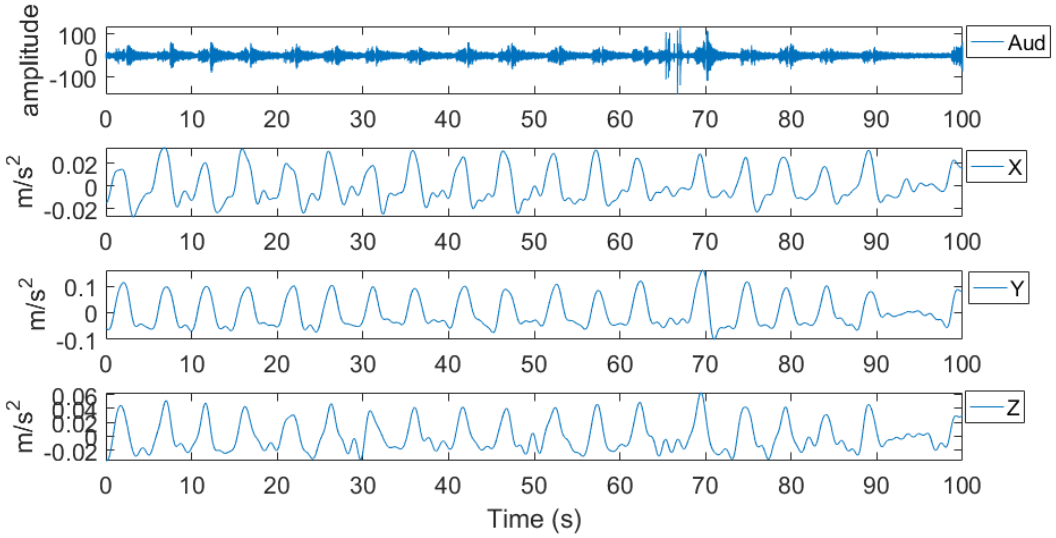


Fig. D.3: Segment of all signals captured from a sleeping family member in the preliminary test. This was to show how the audio acquisition sometimes resulted in very low amplitudes of the signal.

HR and RR Estimation

This section shows the RR and HR estimations in the initial tests conducted on family and friends of the author.

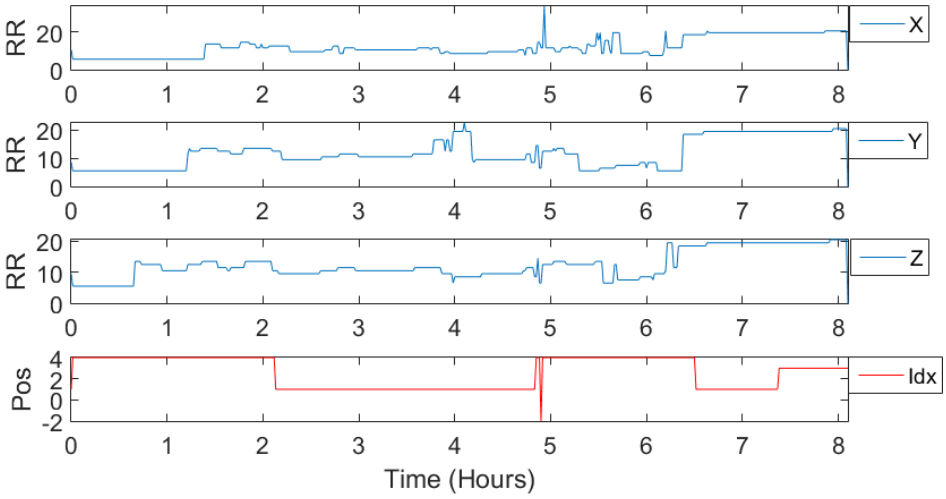


Fig. D.4: Respiration rate estimation with a test person in the initial trials, before conducting the experiment. X, Y and Z are accelerometer readings in each axis, and idx is the sleep position index.

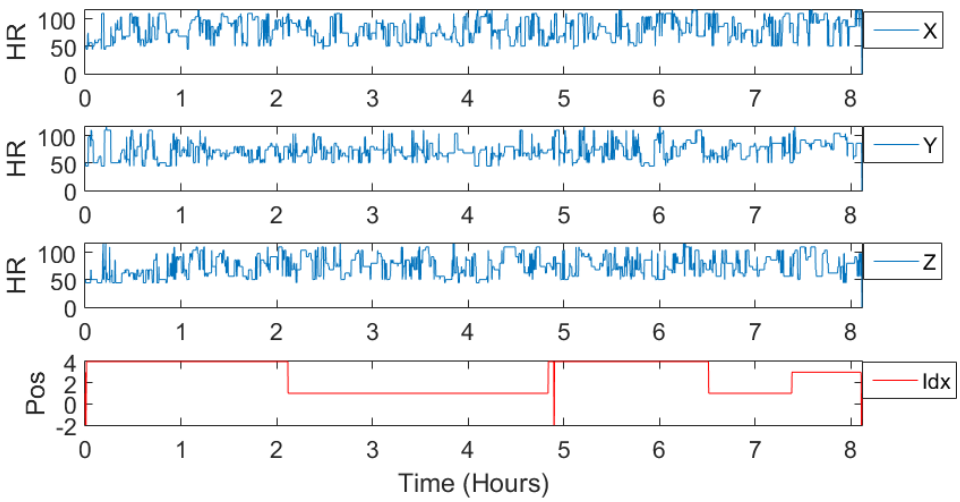


Fig. D.5: Heart rate estimation with a test person in the initial trials, before conducting the experiment. X, Y and Z are accelerometer readings in each axis, and idx is the sleep position index.

Event Detection, Method 1

Only the event detection method 1 was used in the preliminary tests, since the second method was implemented after the clinical experiments had begun.

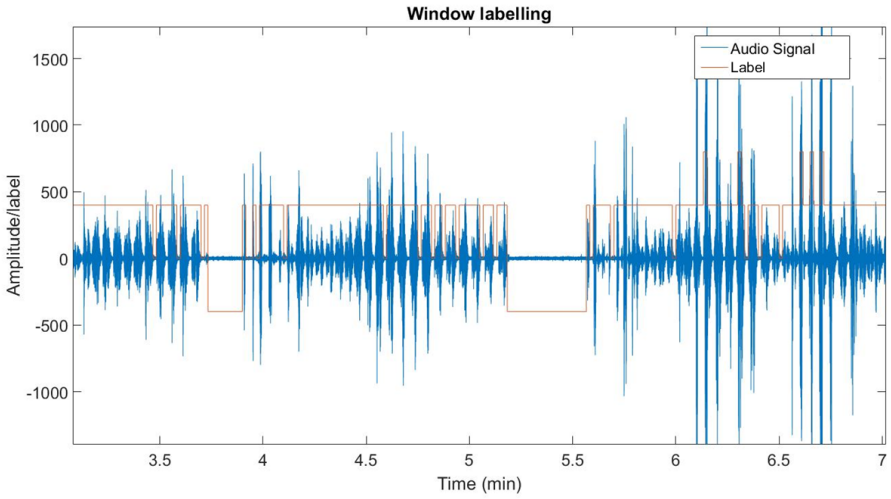


Fig. D.6: Initial test of event detection method 1, from an audiosegment of a nights recording of the author himself. The blue graph is the audio values, and the red lines are the corresponding labels from the event_{m1} method.

D.2 Filters: Phase and Magnitude Response

This section contains all of the phase- and amplitude responses from the filters used for the filtering of the audio and accelerometer signals.

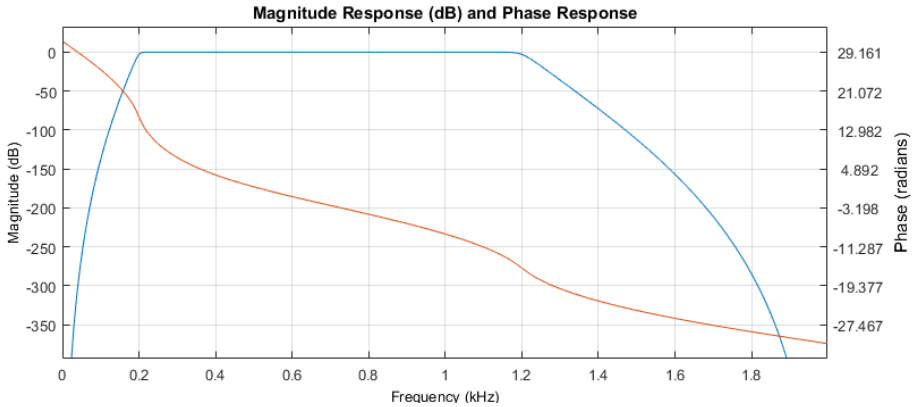


Fig. D.7: Phase and magnitude response of the bandpass filter used for filtering the audio signal, with cutoff frequencies 200 - 1200 Hz.

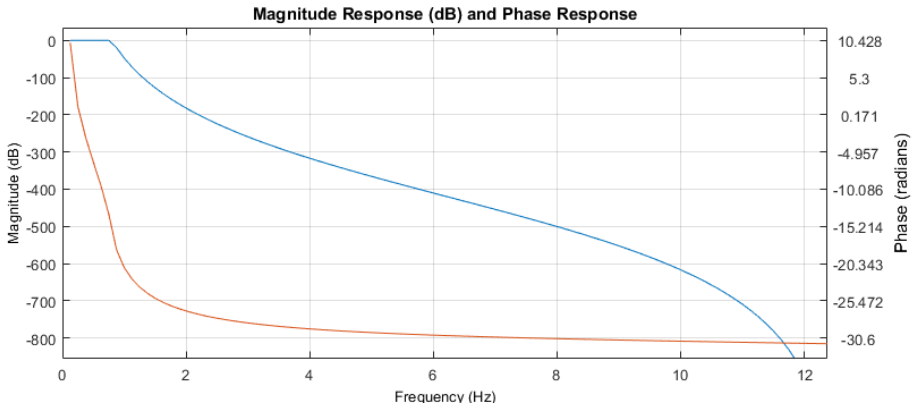


Fig. D.8: Phase and magnitude response of the bandpass filter used for filtering the accelerometer signal for respiration analysis, using cutoff frequencies 0.1 - 0.8 Hz

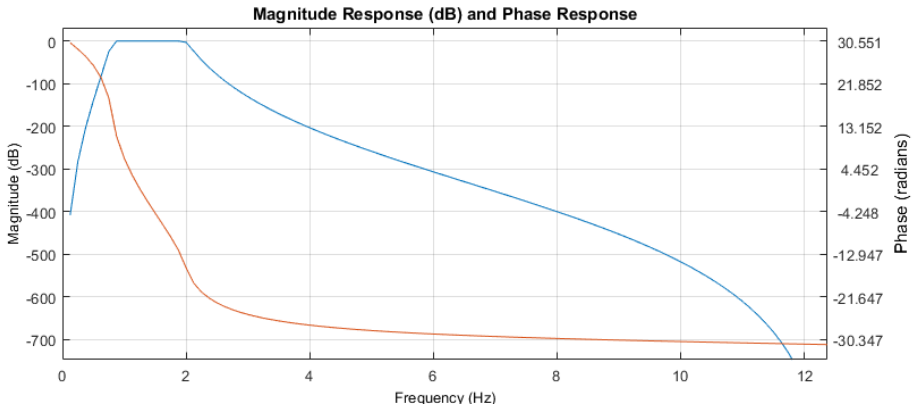


Fig. D.9: Phase and magnitude response of the bandpass filter used for filtering the accelerometer signal for respiration analysis, using cutoff frequencies 0.8 - 2.0 Hz.

D.3 Results; Features

This section includes the results of the feature extraction process

D.3.1 Feature values

Complete table of all features extracted from the dataset consisting of 23 subjects, together with the corresponding means and standard deviations related to each severity class and healthy subjects.

	Feat 1	Feat 2	Feat 3	Feat 4	Feat 5	Feat 6	Feat 7	Feat 8	Feat 9	Feat 10	Feat 11	Feat 12	Feat 13	AHI
Pat 1	8.47	25.18	17.18	10.82	9.00	12.65	9.24	4.24	1530	0.22	23.78	58	2	7.2
Pat 2	15.32	40.70	32.05	24.39	10.78	45.24	35.03	19.22	1269	0.14	31.65	73	3	46.1
Pat 3	1.00	24.52	17.82	10.12	11.41	14.11	8.63	3.56	1263	0.32	37.24	40	4	36.1
Pat 4	3.90	27.67	23.53	19.14	8.53	13.53	9.39	5.12	1477	0.04	41.41	70	3	23.2
Pat 5	15.91	33.48	23.60	18.09	11.55	34.39	28.48	18.73	1403	0.45	31.35	53	1	14.6
Pat 6	33.27	51.75	47.82	42.28	15.88	49.49	45.28	37.89	1558	0.68	25.69	60	3	51.2
Pat 10	0.11	23.39	18.24	10.73	11.59	3.49	1.45	0.43	1678	0.07	37.87	62	5	6.5
Pat 12	35.13	66.60	55.63	44.65	18.01	82.65	71.89	53.29	1230	0.76	32.49	22	1	82.6
Pat 13	0.00	33.95	18.01	6.69	8.92	2.23	0.96	0.00	1129	0.40	25.43	58	2	21.0
Pat 14	1.57	15.00	3.88	2.20	10.75	4.56	3.25	0.84	1717	0.73	24.96	36	2	4.0
Pat 15	10.25	29.86	18.98	10.38	7.85	5.06	2.66	0.13	1422	0.33	28.91	66	2	13.1
Pat 18	1.01	7.14	2.01	0.82	13.19	8.10	3.07	0.64	1966	0.46	20.38	65	3	5.0
Pat 19	25.76	31.55	24.65	14.05	9.86	4.25	2.16	0.06	1460	0.97	21.61	54	3	3.0
Pat 22	13.63	30.97	21.61	19.00	11.70	27.81	21.89	12.39	1308	0.43	27.75	56	3	37.1
Pat 23	6.44	43.45	27.24	14.11	10.46	16.41	11.27	3.22	1454	0.74	27.97	67	3	32.0
Pat 24	7.78	29.20	20.10	10.86	9.02	14.67	9.46	4.48	1227	0.07	29.40	63	3	25.8
Pat 25	1.31	32.60	24.39	17.73	9.40	6.43	3.93	0.77	1513	0.42	26.06	44	3	4.0
Pat 26	14.30	37.68	35.99	31.22	11.61	40.22	33.60	23.15	1170	0.21	31.56	75	2	40.0
Pat 29	11.59	38.35	19.02	8.92	12.71	36.11	22.14	5.94	1211	0.99	34.94	61	4	59.8
Pat 30	2.17	1.82	0.46	0.34	12.88	1.25	0.28	0.00	1580	0.75	22.99	59	6	2.6
Pat 31	13.54	42.55	40.99	29.73	11.57	33.50	27.09	19.54	1502	0.08	31.44	71	1	42.4
Pat 32	2.34	28.28	19.87	11.22	7.48	7.24	4.91	1.17	1540	0.00	22.88	72	2	16.3
Pat 33	36.32	38.25	32.26	32.90	9.72	5.66	2.24	0.64	1685	0.55	31.35	31	2	5.3
μ Healthy (5)	6.36	17.62	11.08	7.03	11.21	4.92	2.54	0.46	1647	0.67	23.20	51.60	3.40	3.72
μ Mild (4)	13.79	29.17	21.66	16.21	9.54	6.71	3.90	1.36	1579	0.29	30.48	54.25	2.75	8.03
μ Mod (5)	3.50	29.77	20.38	11.98	8.49	9.42	6.18	2.69	1343	0.13	29.78	65.75	2.50	20.18
μ Sev (9)	16.02	41.84	33.13	24.93	12.68	38.39	30.76	19.80	1329	0.48	31.19	58.33	2.67	47.48
σ Healthy (5)	10.85	14.01	12.33	8.22	1.73	2.57	1.41	0.40	203	0.23	2.34	11.63	1.52	0.94
σ Mild (4)	15.66	6.64	7.10	11.13	1.57	4.06	3.60	1.93	127	0.20	5.85	15.84	1.50	3.47
σ Mod (5)	3.27	2.85	2.30	5.20	0.71	5.80	4.08	2.49	197	0.18	8.20	6.45	0.58	4.03
σ Sev (9)	11.26	12.06	13.20	13.14	2.55	20.45	19.25	16.76	139	0.32	3.63	17.32	1.12	15.65

D.4 RESULTS Classification

This section has the tables with the performance measures with different feature combinations. Due to limited time, the results was not analyzed, and therefore placed in the appendix.

	Bagged Decision Tree		Support Vector Machine	
Method	**Acc** (%)	**Sens** (%)	**Acc** (%)	**Sens** (%)
Binary, TH = 5	91.3	100	91.3	100
Binary, TH = 10	73.9	62.5	87.0	100
Binary, TH = 15	82.6	80	78.3	100
4-Class	56.5	60	56.5	75

Table D.1: Classification performance using only features from the event detection algorithms $\text{MAS}_{\text{AHI},1}$ and $\text{MAS}_{\text{AHI},2}$.

	Bagged Decision Tree		Support Vector Machine	
Method	**Acc** (%)	**Sens** (%)	**Acc** (%)	**Sens** (%)
Binary, TH = 5	82.6	100	86.9	75
Binary, TH = 10	65.2	50	87.0	71.4
Binary, TH = 15	65.2	75	65.2	62.5
4-Class	47.8	50	39.1	66.7

Table D.2: Classification performance using all features EXCEPT features from the event detection algorithms $\text{MAS}_{\text{AHI},1}$ and $\text{MAS}_{\text{AHI},2}$.

D.5 Visualization tool

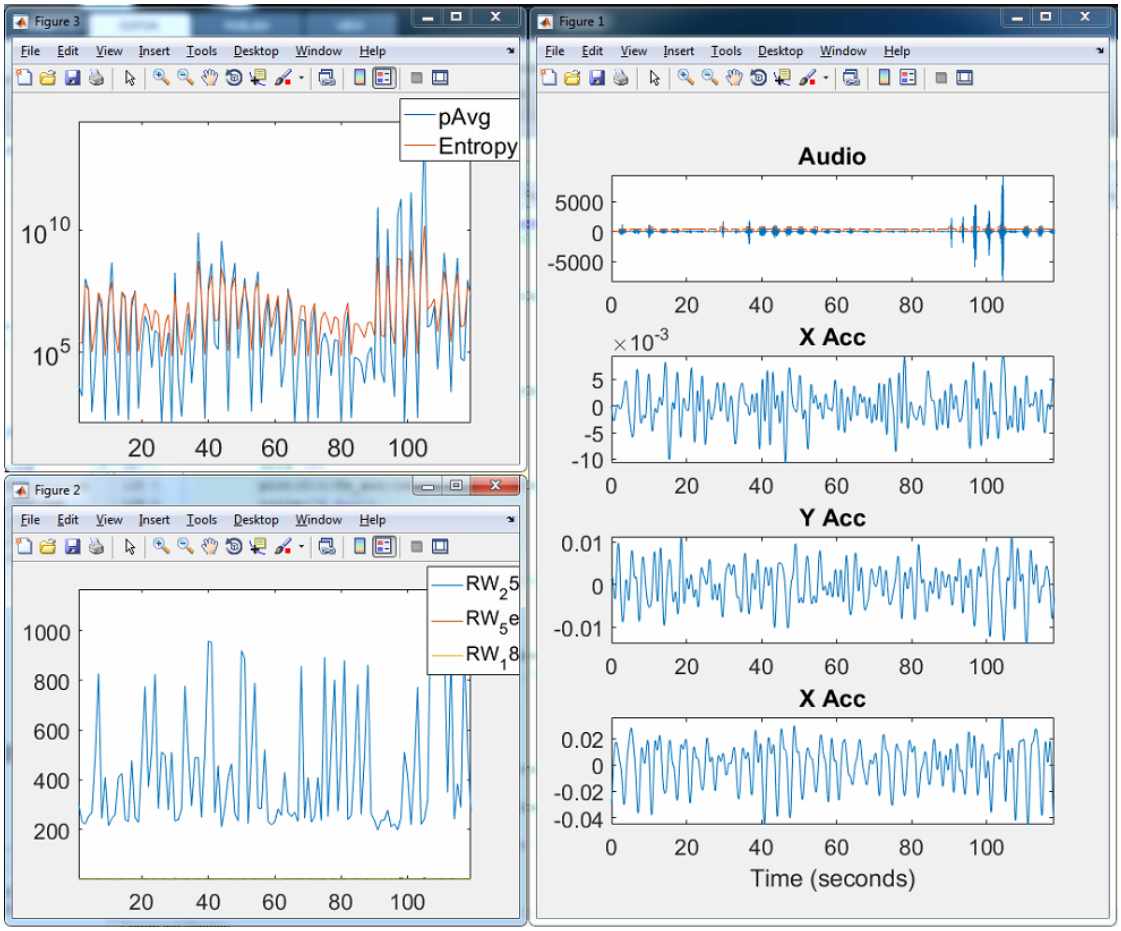


Fig. D.10: Illustration of how it looks when using the visualization tool. The figure 1 (to the right) is the acquired signals which is always shown. The two other figures (to the left) can be modified to show whatever the user thinks is important. In the MATLAB command prompt, it will say what type of *apneic* event there is shown in the window.

E Appendix: CRM Event Extract, Manual

The reason for developing a manual, was to make sure that the users of the visualization tool, new how to extract the necessary information from the CRM data in DOMINO into MATLAB.

Extracting events from CRM analysis

This guide is about extracting information regarding events from a CRM file after it has been analyzed by the medical staff.

Step 1) Start DOMINO and press

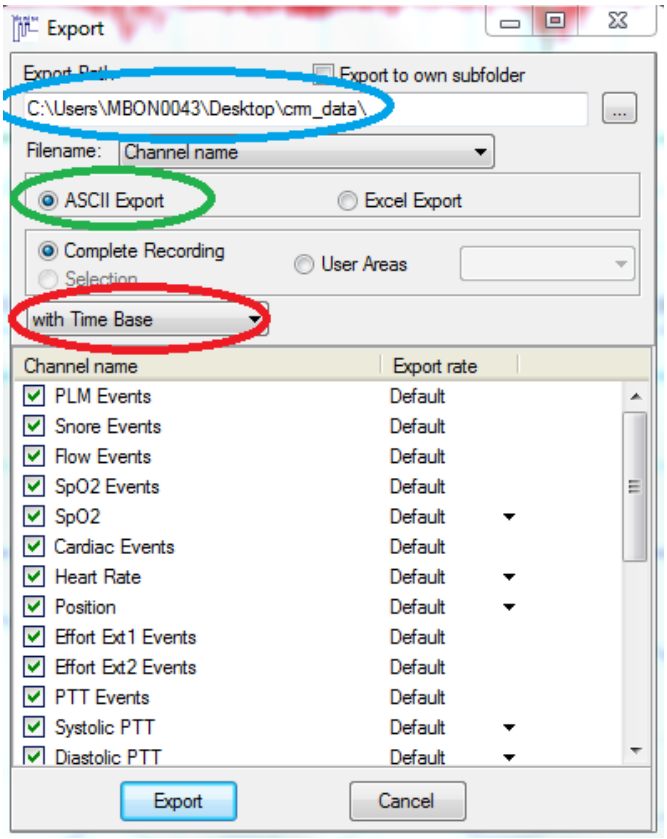
and, double click the patient from where the data will be extracted from.

Step 3) Extract Data

Go to “File” -> “Export Analysis Data”

-> “Data” and the window to the left will be shown.

Specify filepath (blue circle), export file type (green circle, has to be on *ASCII*) and time base (red circle, has to be on *Time Base*). Press Export.



Step 4) Gather events into one text file

This is done by using the MATLAB function *EDF_event_extract(filepath),* where filepath is the filepath specified in the above step (blue circle). The function will iterate through .txt files, extracting event information from each file with events.

The output is a text file names “*event_matrix.txt”*, which will be saved in the same repository as the other text files. Information regarding each events includes; Event type, start time (hh:mm:ss:ms), stop time (hh:mm:ss:ms) and length (sec.). Each event is separated with a space in the text file.

F Appendix: Matlab Code

This appendix contains the MATLAB code used for processing the acquired data. The first subsection contains the main script and the functions used to extract the features. The second subsection contains the main script used for classification and two functions, one for each classifier. The third and last subsection is the scripts used for extracting information from the CRM analysis and a script for visualizing annotated events in the MAS acquired signals.

F.1 Load signal, preprocess and extract features

This section includes the main scrip for loading data, preprocess it, extract features and save it all into matrices ready for classification. All of the functions used to do this, is also in the Appendix.

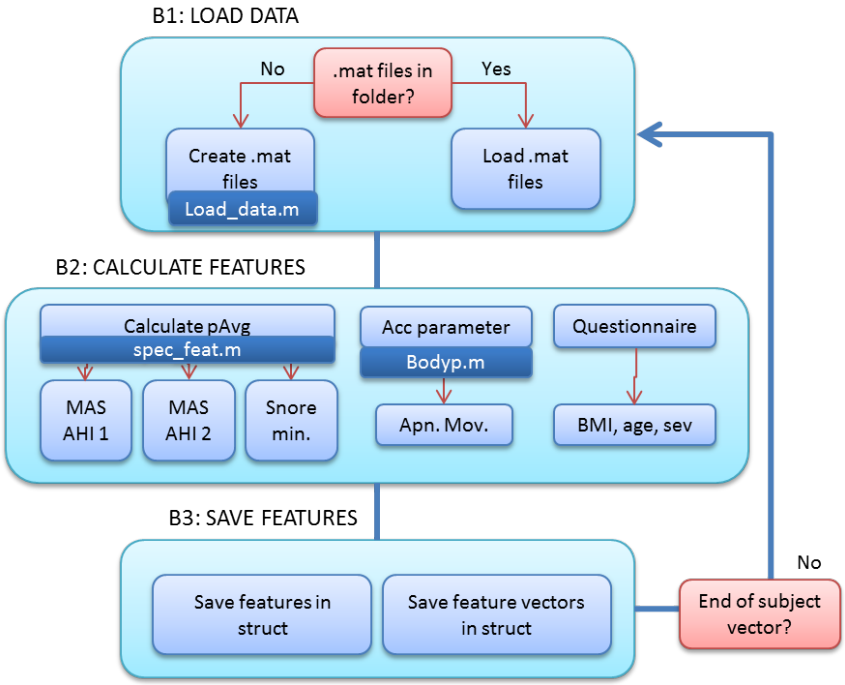


Fig. F.1: The flowchart shows the progression of the main classification algorithm, Algorithm_MAIN_SCRIPT_v2.m, where dark blue boxes are the functions performing the process of the box it is attached to. The top red box indicates that the algorithm checks if a subject have been processed before. The 2nd red box indicates that the algorithm loops through all subjects.

F.1.1 Algorithm_MAIN_SCRIPT_v2.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Main script that loads the data, performs preprocessing and stores the
% data ready for feature extraction. Then it extracts features and saves
% them into a feature matrices. Also loads data from CRM analysis
%
%   Loads: audio, X, Y, Z, fs_acc and fs_aud
%
%   Outputs:    feat_struc:         Struct with features for each patient
%               all_feat:           Vectors used to extract features
%
%   Author: Mathias P. Bonnesen, s113918, Technical University of Denmark.
%
%   Date: 01/05 - 2016
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% SELECT FOLDER AND LOAD DATA %%%
Patvec = {'Pat_1';'Pat_2';'Pat_3';'Pat_4';'Pat_5';'Pat_6';'Pat_8';...
    'Pat_9';'Pat_10';'Pat_12';'Pat_13';'Pat_14';'Pat_15';'Pat_16';...
    'Pat_18';'Pat_19';'Pat_22';'Pat_23';'Pat_24';'Pat_25';'Pat_26';...
    'Pat_29';'Pat_30';'Pat_31';'Pat_32';'Pat_33'}

Patvec = {'Pat_32'}

MODE = 'no'

% Loop through each patient form Patvec
for k = 1:size(Patvec,1)
    Pat = Patvec{k,:};
```

```
main_folder = '\\10.230.149.201\MobilApnea\MATLAB';  % MAIN FOLDER
cd(main_folder)

folder = ['\\10.230.149.201\MobilApnea\DATA\'  Pat]; % Files location

% Look if patient have been processde previously
if isempty(dir([folder '\matlab_data' '\*mat'])) == 1
    disp([Pat ': No .mat files. Creating .mat files...'])

    % Load data
    tic
    [X,Y,Z,fs_acc,audio,fs_aud] = load_data(folder);
    toc

    % Load CRM data to find start and stop
    crm_fold = ['\\10.230.149.201\MobilApnea\DATA\' Pat '\crm'];
    [LAG] = CRM_data_extract(crm_fold); % Find lag between MAS/CRM and
                                        % create event_matrix

    cd([folder '\crm'])

    % Segment to extract Time window segment
    fileID = fopen('event_matrix.txt','r');        % read whole file
    textfile = textscan(fileID,'%s','Delimiter','\n');
    textfile = textfile{:};
    cellf_start = strsplit(textfile{2},';');
    cellf_stop = strsplit(textfile{1},';');
    time_start = cellf_stop{1};
    time_stop = cellf_start{1};
    crm_start = str2num(time_start(24:25))*3600 + str2num(time_start(27:28))*60 ...
        ...
        + str2num(time_start(30:31)) - 24*3600;
    rec_length = str2num(time_stop(11:12))*3600+str2num(time_stop(14:15))*60 ...
        + str2num(time_stop(17:18)) - crm_start;

    % This ensures that the MAS analysis stops when "lights is set ON"
    rec_interval = length(-LAG:rec_length);  % Length in seconds

    % Address inconsistencies in data of Pat 1 and Pat 24
    if strcmp(Pat,'Pat_1') == 1
        rec_interval = length(600:25890); % Adjust clock-miss-set on Pat 1
        LAG = 0;
    elseif strcmp(Pat,'Pat_24') == 1
        rec_interval = length(600:length(audio)/fs_aud); % Take whole signal post ...
            10min
        LAG = 0;
    end

    start = 0;  % Time of analysis start.

    if length(-LAG:rec_length) > length(X)/fs_acc
        X = X(fs_acc*start+1:end); Y = Y(fs_acc*start+1:end); ...
        Z = Z(fs_acc*start+1:end); audio = audio(fs_aud*start+1:end);
    else
        X = X(fs_acc*start+1:rec_interval*fs_acc); Y = Y(fs_acc*start+1:...
            rec_interval*fs_acc); ...
        Z = Z(fs_acc*start+1:rec_interval*fs_acc); audio = audio(fs_aud*start+1:...
            rec_interval*fs_aud);
    end

    cd(main_folder)

    %%%%%%%%%%%%%%%%%%%%%%%%
    % PREPROCESSING
    %%%%%%%%%%%%%%%%%%%%%%%%
    disp('Stating preprocessing of signals...')
    % Audio preprocessing
    tic

    % AUDIO
    Fpass1 = 200;         % First Passband Frequency
    Fpass2 = 1200;         % Second Passband Frequency
    order = 40;
```

```
    d = designfilt('bandpassiir','FilterOrder',order, ...
         'HalfPowerFrequency1',Fpass1,'HalfPowerFrequency2',Fpass2, ...
         'SampleRate',fs_aud);

    audio_p=filtfilt(d,audio);

    % ACCELERATIONS
    Fpass1 = 0.10;   Fpass2 = 0.80;        % Passband Frequencies, - RESP
    Fpass1h = 0.8;   Fpass2h = 2.0;        % Passband Frequencies, - HEART
    order = 40;

    d2 = designfilt('bandpassiir','FilterOrder',order, ...
        'HalfPowerFrequency1',Fpass1,'HalfPowerFrequency2',Fpass2, ...
        'SampleRate',fs_acc);

    X_f=filtfilt(d2,X);
    Y_f=filtfilt(d2,Y);
    Z_f=filtfilt(d2,Z);

    % Filter for HEART frequency
    d3 = designfilt('bandpassiir','FilterOrder',order, ...
        'HalfPowerFrequency1',Fpass1h,'HalfPowerFrequency2',Fpass2h, ...
        'SampleRate',fs_acc);

    X_fh=filtfilt(d3,X);
    Y_fh=filtfilt(d3,Y);
    Z_fh=filtfilt(d3,Z);

    info = [fs_acc;fs_aud;LAG+start];

    posvec = bodyp(X,Y,Z);  % Position is calculated here, since it
                            % the raw accelerometer data.

    % Save procesed signals
    cd([folder '\matlab_data'])
    save('X_f','X_f'); save('Y_f','Y_f'); save('Z_f','Z_f');
    save('X_fh','X_fh'); save('Y_fh','Y_fh'); save('Z_fh','Z_fh')
    save('posvec','posvec'); save('audio_p','audio_p')
    save('info','info')

    cd(main_folder)
else
    % If Pat have previosly been precossed, just load files.
    disp([Pat ': .mat files located. Loading...'])
    cd([folder '\matlab_data'])
    % Load the cut_signal matfile
    load('X_f','X_f'); load('Y_f','Y_f'); load('Z_f','Z_f');
    load('X_fh','X_fh'); load('Y_fh','Y_fh'); load('Z_fh','Z_fh')
    load('posvec','posvec'); load('audio_p','audio_p')
    load('info','info')
    fs_acc = info(1); fs_aud = info(2); LAG = info(3);
    cd(main_folder)
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FEATURE EXTRACTION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
disp('Feature extraction beginning...')

if strcmp(MODE,'surpine') == 1
    posvec_a = ones(1,floor(length(audio_p)/fs_aud));
    for tv = 2:length(X_f)/fs_acc-1
       posvec_a(tv*fs_aud:(tv+1)*fs_aud-1) = ...
            ones(1,fs_aud)*posvec((tv-1)*25);
    end

    disp('Surpine calculations')
    X_f = X_f(posvec==1); Y_f = Y_f(posvec==1); Z_f = Z_f(posvec==1);
    audio_p = audio_p(posvec_a==1);
end

%%%%% Computing the Signal Vector Magnitude %%%%%
```

```
    c_acc = sqrt(X_f.^2+Y_f.^2+Z_f.^2);  % Composite accelerations for Actiography

    % Label windows
    w = 1;                                                  % Window size.
    ov = 1;                                                 % Overlap [0:1]
    a_window = 0;                                           % Was 1800 in prior calc


    %%% Feature vectors %%%
    pAvg_vec = zeros(1,length(a_window:1/w:length(audio_p)/fs_aud));
    se_vec = zeros(1,length(a_window:1/w:length(audio_p)/fs_aud));
    ZCR_vec = zeros(1,length(a_window:1/w:length(audio_p)/fs_aud));
    mov_vec = zeros(1,length(a_window:1/w:length(audio_p)/fs_aud));
    rest_vec = zeros(8,length(a_window:1/w:length(audio_p)/fs_aud));
    posvec_sec = zeros(1,length(a_window:1/w:length(audio_p)/fs_aud));

    %%% Label vectors
    lab_vec2 = zeros(1,length(a_window:1/w:length(audio_p)/fs_aud));

    % Calculate spectral features from window.
    for i = 1:ov:length(audio_p)/fs_aud - 5
        au_seg = audio_p(i*fs_aud:i*fs_aud+fs_aud*w);
        acc_seg = i*fs_acc:i*fs_acc+fs_acc*w;

        [pAvg,fp,se,RW_25,RW_5e,RW_18,RB_1,RB_2,skew,kurt]...
            = spec_feat(pwelch(au_seg*w,fs_aud*w));

        % mov_T
         mov = quantile(c_acc(acc_seg),0.50);              % Quantile

         % Zero-croos-rate
        ZCR=mean(abs(diff(sign(au_seg))));

        pAvg_vec(i) = pAvg;
        se_vec(i) = se;
        ZCR_vec(i) = ZCR;
        mov_vec(i) = mov;
        rest_vec(:,i) = [fp,RW_25,RW_5e,RW_18,RB_1,RB_2,skew,kurt];
    end

%%%% Calculate Features %%%%%
    %%% EVENT METHOD 1 %%%
    MAS_AHI = zeros(1,4);
    cs = 1;
    w = 1; ov = 1;
    disp('Labelling windows...')

    for ahi_t = [5*10^3,quantile(pAvg_vec,0.60),quantile(pAvg_vec,0.50),...
            quantile(pAvg_vec,0.40)]
        lab_vec = ones(length(audio_p),1);   % Label vector
        event_vec = zeros(length(audio_p),1);

        c = 0;
        c_val2 = 0;
        c_val = 0;

        for i = 1:length(pAvg_vec)
            pAvg = pAvg_vec(i);

             if (pAvg)*w < (ahi_t)*w %|| (pAvg)*w < se %RW_25 < 0.5%&& se < seT/10
               lab_vec(i*fs_aud:i*fs_aud+fs_aud) = zeros(length(fs_aud),1);
               c_val = c_val + 1;
               if c_val > 9*(1/(w*ov))
                    event_vec(i*fs_aud-9*fs_aud:i*fs_aud+fs_aud)=ones(length(fs_aud),1)↩
                        *1;
                    lab_vec(i*fs_aud-9*fs_aud:i*fs_aud+fs_aud) = ones(length(fs_aud),1)↩
                        *(-1);
                    c = c + 1;
               end
            elseif pAvg > 10^8 % && se > seT*100
               lab_vec(i*fs_aud:i*fs_aud+fs_aud) = ones(length(fs_aud),1)*2;
               if c_val > 9; c_val2 = c_val2 + 1; end
               c_val = 0;
```

```
            else
                lab_vec(i*fs_aud:i*fs_aud+fs_aud) = ones(length(fs_aud),1);
                if c_val > 9; c_val2 = c_val2 + 1; end
                c_val = 0;
            end

        lab_vec2(i) = unique(lab_vec(i*fs_aud:i*fs_aud));
    end

    %%% Event Method 1 - finds apneic moments
    MAS_AHI(cs) = c_val2/(length(X_f)/(fs_acc*3600));
    cs = cs+1;
end


%%%% Event Method 2 %%%
    sam_vec_pAvg = zeros(length(pAvg_vec),1);
    event_vec_pAvg = zeros(length(pAvg_vec),1);

    for sample = 20:length(sam_vec_pAvg)-11
        sumAf = sum(log(pAvg_vec(sample:sample+9)))/10;
        sumBe = sum(log(pAvg_vec(sample - 10:sample-1)))/10;
        sam_vec_pAvg(sample) = sumAf/sumBe;
    end

    disp('Calculating MAS_AHI type 2...')
    MAS_AHI2 = zeros(1,4);
    mc = 1;
    for Thr = [quantile(sam_vec_pAvg,0.99),1.8,2,2.5]
        loopvec = event_vec_pAvg;
        [pks,locs] = findpeaks(sam_vec_pAvg); % Find peaks aboive 1.8 Threshold
        loopvec(locs(pks>Thr)) = 0.5;

        % Implement min peaks distance on the threshold-peaks
        [pks,locs] = findpeaks(loopvec,'MinPeakDistance',9);
        loopvec(locs(diff(locs)>9)) = 1;
        loopvec(loopvec==0.5) = 0;

        % Calculate MAS_AHI
        MAS_AHI2(mc) = sum(abs(loopvec))/(length(pAvg_vec)/3600);
        mc = mc + 1;
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% QUESTIONAIRE features %%%
filename = [folder,'\iFile.txt'];
ifile_ID = fopen(filename,'r');         % read whole file
textfile = textscan(ifile_ID,'%s','Delimiter','\n');
textfile = textfile{:};

% BMI
bmi_answ = strsplit(textfile{1},';');
bmi_answ = bmi_answ{:};
bmi = (str2num(bmi_answ(20:end-1)))/(str2num(bmi_answ(8:10))/100)^2;

% Rest of answers
answ = strsplit(textfile{5},';');
answ = answ{1};
Q_answ = [];
for s = [20,27,30,33,36,39,42]
        val = str2num(answ(s))-1;
        Q_answ = [Q_answ; val];
end
% Save age into variable
Age = str2num(answ(23:24));
% Calculate OSA severity based on Q's
Q_answ(2) = fix(str2num(answ(23:24))/50); % 'fix' ignores decimal val
Q_sev = sum(Q_answ) + fix(bmi/35);


%%% Apnea Movement minutes %%
```

```
    c_acc_s = zeros(1,ceil(length(c_acc)/fs_acc));
    cs = 1;

    for ii = 1+fs_acc:fs_acc:length(c_acc)
        c_acc_s(cs) = rms(c_acc(ii-fs_acc:ii));
        cs = cs+1;
    end

    mov_T = quantile(c_acc_s,0.95);              % Accelerometer Thresholds
    apn_mov = length(c_acc_s(c_acc_s>mov_T));

    %%% Percentage in surpine position
    surpine = sum(posvec==1)/length(posvec);

    %%% Position in seconds
    posvec_sec = posvec(1:25:end);

    % Respiration rate analysis
    w = 30; % Window size for analysis
    [rr_vecX,rr_vecY,rr_vecZ] = rr_analysis(X_f,Y_f,Z_f,fs_acc,w);
    rrQy = quantile(diff(rr_vecY),[0.10 0.90]);  % Thresholds
    rrQz = quantile(diff(rr_vecY),[0.10 0.90]);  % Thresholds
    RR_peaks = [sum(diff(rr_vecY)<rrQy(1))+ sum(diff(rr_vecY)>rrQy(2)) ...
        sum(diff(rr_vecZ)<rrQz(1))+ sum(diff(rr_vecZ)>rrQz(2))];

    % Snore variability analyzes snoring
    snores_pAvg = log(pAvg_vec(lab_vec2==2));   % LOG of snore power

    % Apnea snoring minutes
     sn_thresh = 900;
     sn_min = sum(rest_vec(1,:)>900);

    % remove a field:
    %feat_struc = rmfield(feat_struc,'Pat_25')

    %%%%%%%%%%%%%%%%%%%%%%%%%
    % SAVE INTO Feature Vectors
    %%%%%%%%%%%%%%%%%%%%%%%%%

    if strcmp(MODE,'surpine') == 0
        load('FEATURES_sctruct.mat')       % Load feature structure
        load('ALL_FEATURES','all_feat')

        feat_struc.(Pat) = [MAS_AHI MAS_AHI2 apn_mov sn_min surpine bmi Age Q_sev]
        all_feat.(Pat) = [pAvg_vec;se_vec;rest_vec;event_vec_pAvg'];

        save('ALL_FEATURES','all_feat')
        save('FEATURES_sctruct','feat_struc')
    else
        load('FEATURES_sctruct_surp.mat')       % Load feature structure
        load('ALL_FEATURES_surp','all_feat')

        feat_struc.(Pat) = [MAS_AHI MAS_AHI2 apn_mov sn_min surpine bmi Age Q_sev]
        all_feat.(Pat) = [pAvg_vec;se_vec;rest_vec;lab_vec2;event_vec_pAvg'];

        save('ALL_FEATURES_surp','all_feat')
        save('FEATURES_sctruct_surp','feat_struc')
    end

    % posvec_sec; has been removed
    disp([Pat '... Done'])

end
```

F.1.2 load_data.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function loads the data into variables.
%
```

```
% Input:  folder           = Folder containing the data
%
% Output: X_rs,Y_rs,Z_rs   = Accelerations in x,y,z axis, resampled to take
%                            take into account the sampling freq delay.
%         fs_acc           = accelerometer sampling frequency
%         audio            = downsampled audio signal
%         fs_aud           = audio sampling frequency
%
%   Author: Mathias P. Bonnesen, s113918, Technical University of Denmark.
%
%   Date: 12/23 - 2016
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [Xr,Yr,Zr,fs_acc,audio,fs_aud,X,Y,Z] = load_data(folder)
current_folder = cd;
cd(folder)

% LOAD AUDIO-data
fid = fopen('audio.pcm','r'); % open stream
speech = fread(fid, inf, 'short', 0, 'l'); % read stream
fclose(fid); % close stream
fs_aud = 16E3; % sampling frequence

% Downsample
ds_f = 4;                           % Downsampling Factor
audio = decimate(speech,ds_f);
fs_aud = fs_aud/ds_f;

% LOAD ACC-data
xFile = 'xFile.txt'; yFile = 'yFile.txt'; zFile = 'zFile.txt';
tFile = 'tFile.txt';
delimiter = ',';

X = dlmread(xFile,delimiter);  % Store in x-vector
Y = dlmread(yFile,delimiter);  % Store in y-vector
Z = dlmread(zFile,delimiter);  % Store in z-vector
T = dlmread(tFile,delimiter);  % Store in T-vecPator

%%% RESAMPLE OF ACC SIGNALS
%%
T_s = T - T(1);                             % Start timestamp vector at 0
T_sn = T_s/10^9;                            % Transform to seconds

tvec_res = 0:1/25:length(speech)/fs_aud'; % Define time axis
fs_acc = 25;                                % Resampled FS

[Xr,tx] = resample(X,T_sn,fs_acc,40,80);
[Yr,ty] = resample(Y,T_sn,fs_acc,40,80);
[Zr,tz] = resample(Z,T_sn,fs_acc,40,80);

cd(current_folder) % Return to main folder
end
```

F.1.3 spec_feat.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculates a number of spectral features from an input
% discrete signal.
%
% INPUT:  sig        = Power Spectral Density of signal
%
% OUTPUT: pAvg       = Average Power
%         zcr        = Zero Cross Rate
%         f0         = Spectral peak, low
%         fp         = Spectral peak, high
%         se         = Spectral entropy
%
```

```
%   Author: Mathias P. Bonnesen, s113918, Technical University of Denmark.
%
%   Date: 12/23 - 2016
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ pAvg,fp,se,RW_25,RW_5e,RW_18,RB_1,RB_2,skew,kurt ] = spec_feat(sig)

% Average Power pAvg
pAvg = mean(sig.^2);

% Frequency of the spectral peak with the maximum power frequency
fp = find(sig==max(sig));   % where t is the frequency axis
if length(fp) > 1
    fp = 0;
end

%%% Spectral entronpy %%%
sig_pdf = sig/sum(sig);     % Normalize PSD so it can be viewed as a pdf
se = -sum(sig.*log(sig_pdf));

%%% Ratio of Energy bands
RW_25 = sum(sig(200:500))/sum(sig); % Energy in frq. 200-500 div. by total
RW_5e = sum(sig(500:end))/sum(sig); % Energy in frq. 200-500 div. by total
RW_18 = sum(sig(1:800))/sum(sig);   % Energy in frq. 0-800 div. by total

RB_1 = sum(sig(200:500))/...
    (sum(sig)-sum(sig(200:500)));  % Energy in frq. 200-500 div. by rest

RB_2 = sum(sig(1:800))/...
    (sum(sig)-sum(sig(1:800)));  % Energy in frq. 1-800 div. by rest

%%% Skewness and Kurtosis
skew = skewness(sig);           % Caluclate the skewness of the signal
kurt = kurtosis(sig);           % Calculate the kurtosis of the signal

end
```

F.1.4 bodyp.m

```
function [posVec] = bodyp(X,Y,Z)
% "bodyp_func" is a function that calculates the
% current body posture using accelerometer data.
%
% INPUT: X,Y,Z         = Vectors with accelerations
%                        in X, Y and Z axis.
%
% OUTPUT: postVec       = Vector with positions for
%                         each timestep.
% Original code in JAVA: Casper B. Jespersen, Mads
%                        Olsen & Steen Lillelund
%
%   MATLAB: Mathias P. Bonnesen, s113918, Technical University of Denmark.
%
%   Date: 12/23 - 2016
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% INFO Regarding positions
% Position:  1: Surpine
%            2: Prone
%            3: Left lateral
%            4: Right lateral
%            5: Nothing to declare
%           -2: upright
%%%
%%% INITIATION ###
init = 0;           % 0 = false, 1 = true
w = 5;              % Length of a window
```

```
posVec = [];        % Vector with positions
c = 1;               % Counts

%%% GENERATE DATA STRUCTURE %%%
C_acc_data = zeros(3,length(X));
C_acc_data(1,:) = X;
C_acc_data(2,:) = Y;
C_acc_data(3,:) = Z;

for k = 1:length(X)-w
    acc_data = C_acc_data(:,k:k+w);

% Moving average window
    if init == 0
        x_sum = 0;
        y_sum = 0;
        z_sum = 0;
        init = 1;

        % Sum of values inside the window
        for ii = 1:w
            x_sum = x_sum + acc_data(1,ii);
            y_sum = y_sum + acc_data(2,ii);
            z_sum = z_sum + acc_data(3,ii);
        end

        % Window Average
        x_sum = x_sum/w;
        y_sum = y_sum/w;
        z_sum = z_sum/w;

        if abs(x_sum) > abs(y_sum) && abs(x_sum) > ...
                abs(z_sum)
            if x_sum < 0
                statusposition = 3; %% left lateral
            else
                statusposition = 4; %% right lateral
            end
        elseif abs(y_sum) > abs(x_sum) && abs(y_sum) > ...
                abs(z_sum)
            statusposition = -2; % upright
        elseif abs(z_sum) > abs(x_sum) && abs(z_sum) > ...
                abs(y_sum)
            if z_sum > 0
                statusposition = 1; % Surpine
            else
                statusposition = 2; % prone
            end
        else
            statusposition = 5; % Nothing to declare
        end
    end
    posVec(c) = statusposition;
    c = c + 1;
    init = 0;
end

end
```

F.1.5 PlotfSpec.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Simple function to illustrate the frequency spectrum of a signal
%   INPUTS:
%   sig     =     The decired signal
%   fs      =     The sampling frequency
%
%   OUTPUTS:
%   mFreq   =     Maximum Frequency
```

```
%
%   Author: Mathias P. Bonnesen, s113918, Technical University of Denmark.
%
%   Date: 12/23 - 2016
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [mFreq] = PlotFspec(signal,fs,O)

% Fourier Transform:
x = fftshift(fft(signal));

% Frequency specifications:
N = length(signal);
dF = fs/N;                      % Hz
f = -fs/2:dF:fs/2-dF;           % Hz

% Time axis
t = 0:1/fs:length(signal)/fs-1/fs;

faxis = f(floor(N/2):floor(N*3/4));        % Frequency axis
aaxis = abs(x(floor(N/2):floor(N*3/4)))/N;  % Amplitude axis

% Plotting the spectrum
if strcmp(O,'ON')
    figure(2)
    subplot(2,1,1)
    plot(faxis,aaxis)
    axis tight
    title(sprintf('Segment %i'),'Fontsize',16,'FontWeight','bold')
    xlabel('Frequency [Hz]','Fontsize',15)
    ylabel('Amplitude [mV]','Fontsize',15)
    subplot(2,1,2)
    plot(t,signal)
    xlabel('Time [Sec]','Fontsize',15)
    ylabel('Amplitude [mV]','Fontsize',15)
    set(gca,'Fontsize',15)
end

%% Find maximum amplitude freq
mFreq = faxis(find(aaxis==max(aaxis)));

end
```

F.2 Classification

This section includes all of the code used for classifying the subjects into different classes related to severity of OSA. Below in Fig. F.2, a flowchart of the classification can be seen.

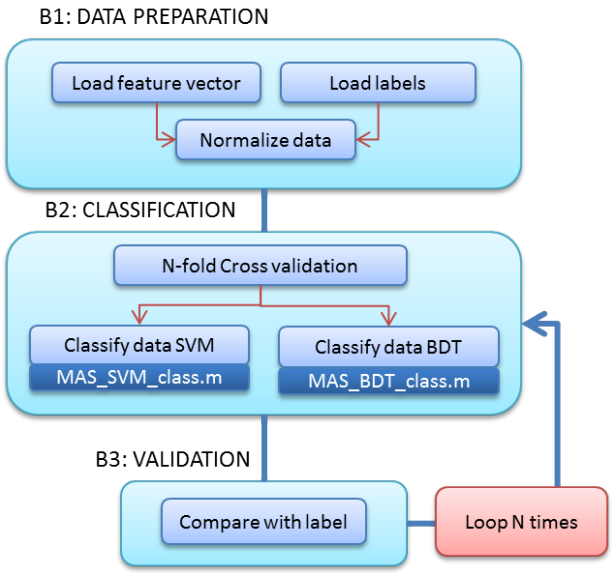


Fig. F.2: Flowchart of how the algorithm performs the classification. The flowchart shows the progression of the main classification algorithm, classification_script.m, and dark blue boxes are the functions used to do the process of the box it is attached to. The red box indicates that the algorithm performs N-fold cross validation. N = 23 since *leave-one-out* crossvalidation was used.

F.2.1 classification_script.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Script which performs CLASSIFICATION and VALIDATION to a dataset.
%
% This script starts by loading the feature vectors and then performs a
% normalization. Afterwards the AHI-indexes for each subject is labelled
% using binary or multi-class labels, and merged with the feature vector
% into one "training_data.mat". This is used for classifying the data using
% a SVM model and Random Forest.
%
% The function (non-MATLAB) created for use in the script are:
% RandomnF_class:       Performs Random Forest classification on a dataset,
%                        and outputs classification results.
% MAS_SVM_class:        Performs SVM classification on a dataset, and
%                       outputs classification results.
%
% Author: Mathias P. Bonnesen, s113918, Technical University of Denmark.
%
% Date: 12/23 - 2016
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% CREATE TRAINING MATRIX %%%
% Load feature structure
disp('Initializing Classification...')
load('FEATURES_sctruct.mat')
% Struct with ID's of all Patients included in the classification
Patvec = {'Pat_1';'Pat_2';'Pat_3';'Pat_4';'Pat_5';'Pat_6';...
    'Pat_10';'Pat_12';'Pat_13';'Pat_14';'Pat_15';...
    'Pat_18';'Pat_19';'Pat_22';'Pat_23';'Pat_24';'Pat_25';'Pat_26'...
    ;'Pat_29';'Pat_30';'Pat_31';'Pat_32';'Pat_33'};

% Assign AHI values to vector
AHI_val = [7.2;46.1;36.1;23.2;14.6;51.2;6.5;82.6;...
    21;4;13.1;5;3.1;37.1;32;25.8;4;40;59.8;2.6;42.4;16.3;5.3]';



AHIC_2 = repmat(AHI_val,3,1);

AHIC_4 = AHI_val;

% Convert values to 2-class labels
thr = [5,10,15];
for i = 1:3
    AHIC_2(i,AHIC_2(i,:)<=thr(i)) = 0;
    AHIC_2(i,AHIC_2(i,:)>thr(i)) = 1;
end

% Convert Values to 4-class labels
AHIC_4(AHIC_4<=5) = 0;
AHIC_4(AHIC_4>30) = 3;
AHIC_4(AHIC_4>15) = 2;
AHIC_4(AHIC_4>5) = 1;

resMat = zeros(4,4);

for it = 1:4
    it

    % Merge the features with the labels
    featmat = zeros(length(Patvec),length(feat_struc.Pat_1)+1);
    %featmat(:,5:end)

    nr_class = 1;

    if it < 4
        AHI_labels = AHIC_2(it,:);
    else
        AHI_labels = AHIC_4;
```

```
    end

    for s = 1:length(Patvec)
        pt = Patvec{s};
        featmat(s,:) = [feat_struc.(pt) AHI_labels(s)];
    end

    % Control which features are used for classification
    %featmat=featmat(:,11:end);

    % General parameters
    classes = [0:max(featmat(:,end))]';
    K = 20;

    %featmat = featmat(:,6:end)

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%% Bagged Decision Tree %%%

    % RF specific parameters
    if nr_class == 1
        nr_trees = 50; % Number of decision trees in the ensamble classifier
    else
       nr_trees = 50;
    end

    % Classification
    [ClassifierRF,AccRF,PredictRF, ScoresRF] ...
        = MAS_BDT_class(featmat,nr_trees,classes,K);

    CMAT_RF = confusionmat (featmat(:,end),PredictRF);

    Acc_RF = trace(CMAT_RF)/sum(sum(CMAT_RF))*100;
    Sens_RF = sum(CMAT_RF(1,1))/sum(CMAT_RF(:,1))*100;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%% SUPPORT VECTOR MACBHINE %%%

    % SVM specific parameters
    [ClassifierSVM, AccSVM, PredictionsSVM, ScoresSVM] ...
        = MAS_SVM_class(featmat,classes,K);

    %[Scores,featmat(:,end)]

    CMAT_SVM = confusionmat (featmat(:,end),PredictionsSVM)

    Acc_svm = trace(CMAT_SVM)/sum(sum(CMAT_SVM))*100;
    Sens_svm = sum(CMAT_SVM(1,1))/sum(CMAT_SVM(:,1))*100;

    resMat(it,:) = [Acc_RF,Sens_RF,Acc_svm,Sens_svm];

     load('Validation_Matrix','validation')
     switch it
         case it == 1
             validation.SC1 = {CMAT_RF,PredictRF;CMAT_SVM,PredictionsSVM}
         case it == 2
             validation.SC2 = {CMAT_RF,PredictRF;CMAT_SVM,PredictionsSVM}
         case it == 3
             validation.SC3 = {CMAT_RF,PredictRF;CMAT_SVM,PredictionsSVM}
         case it == 4
             validation.MC = {CMAT_RF,PredictRF;CMAT_SVM,PredictionsSVM}
         end
     save('Validation_Matrix','validation')
end
```

F.2.2 MAS_SVM_class.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
```

```
%   This function classifies an input matrix containing feature and labels,
%   and makes a Random Forest supervized learning classification on the
%   data. The MATLAB app "Classification Learner" was used to create the
%   frame of the code.
%
%   Input:
%       trainingData:  MxN feature matrix, where M is sample size, and N is
%                      features. Last row of features must be labels.
%       Classes:       Vector with classes, Ex. [0; 1; 2; 3]
%       K:             K-fold cross-validation integer
%
%   Output:
%       Classifier:    Object containing the trained classifier. This can
%                      be used to test on new data. The sub-structure
%                      Classifier.predictFcn is used for this.
%
%       Accuracy:      The accuracy of the trained classifier.
%       Predictions:   The predictions of the data, made by the classifier
%       Scores:        In case of RF or multi-class SVM, this illustrates
%                      the scoring made be the classifiers used to make the
%                      prediction.
%
%   Author: Mathias P. Bonnesen, s113918, Technical University of Denmark.
%
%   Date: 12/23 - 2016
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [Classifier, Accuracy, Predictions, Scores] ...
    = MAS_SVM_class(trainingData,classes,K)

% Read features into matrix
predictors = trainingData(:,1:end-1);

% Assign class labels to response vector
response = trainingData(:,end);


%%% TRAINING THE SVM
% An if-loop determines if its a binary og multiclass classification
% problem
if length(classes) > 2
    % In case of a multi-class problem, multiple SVM's must be classified,
    % , and used for classification (one-vs-one or one-vs-all)
    template = templateSVM(...
        'KernelFunction', 'linear', ...
        'PolynomialOrder', [], ...
        'KernelScale', 'auto', ...
        'BoxConstraint', 1, ...
        'Standardize', true);
    classificationSVM = fitcecoc(...
        predictors, ...
        response, ...
        'Learners', template, ...
        'Coding', 'onevsone', ...
        'ClassNames', classes);
else
    % In case of a binary classification, one SVM is created

    classificationSVM = fitcsvm(...
        predictors, ...
        response, ...
        'KernelFunction', 'linear', ...
        'PolynomialOrder', [], ...
        'KernelScale', 'auto', ...
        'BoxConstraint', 1, ...
        'Standardize', true, ...
        'ClassNames', classes);
end

% Create the result struct with predict function
predictorExtractionFcn = @(x) array2table(x, 'VariableNames', predictorNames);
svmPredictFcn = @(x) predict(classificationSVM, x);
Classifier.predictFcn = @(x) svmPredictFcn(predictorExtractionFcn(x));
```

```
% Add additional fields to the result struct
Classifier.ClassificationSVM = classificationSVM;

% Perform cross-validation
partitionedModel = crossval(Classifier.ClassificationSVM, 'KFold', K);

% Compute validation accuracy
Accuracy = 1 - kfoldLoss(partitionedModel, 'LossFun', 'ClassifError');

% Compute validation predictions and scores
[Predictions, Scores] = kfoldPredict(partitionedModel);
```

F.2.3 MAS_BDT_class.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   This function classifies an input matrix containing feature and labels,
%   and makes a Bagged Decision Tree supervized learning classification on the
%   data.  The MATLAB app "Classification Learner" was used to create the
%   frame of the code.
%
%   Input:
%       trainingData:  MxN feature matrix, where M is sample size, and N is
%                      features. Last row of features must be labels.
%       learners:      Number of learners used in the ensamble method (Int)
%       Classes:       Vector with classes, Ex. [0; 1; 2; 3]
%       K:             K-fold cross-validation integer
%
%   Output:
%       Classifier:    Object containing the trained classifier. This can
%                      be used to test on new data. The sub-structure
%                      Classifier.predictFcn is used for this.
%
%       Accuracy:      The accuracy of the trained classifier.
%       Predictions:   The predictions of the data, made by the classifier
%       Scores:        In case of RF or multi-class SVM, this illustrates
%                      the scoring made be the classifiers used to make the
%                      prediction.
%
%   Author: Mathias P. Bonnesen, s113918, Technical University of Denmark.
%
%   Date: 12/23 - 2016
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [Classifier, Accuracy,Predictions, Scores]...
    = MAS_BDT_class(trainingData,learners,classes,K)

% Extract features into a predictor vector
predictors = trainingData(:,1:end-1);

% Extract labels into a response vector
response = trainingData(:,end);


% Create Cost matrix and assign 2x to increase cost of True Negative
vec1 = ones(1,length(classes));
cost = ones(length(classes),length(classes)) - diag(vec1);
cost(2:length(classes),1) = 2;

% Train a classifier
% This code specifies all the classifier options and trains the classifier.
classificationEnsemble = fitensemble(...
    predictors, ...
    response, ...
    'Bag', ...
    learners, ...
    'Tree', ...
    'Type', 'Classification', ...
```

```
    'ClassNames', classes,'Cost',cost);

% Create the result struct with predict function
predictorExtractionFcn = @(x) array2table(x, 'VariableNames', predictorNames);
ensemblePredictFcn = @(x) predict(classificationEnsemble, x);
Classifier.predictFcn = @(x) ensemblePredictFcn(predictorExtractionFcn(x));

% Add additional fields to the result struct
Classifier.ClassificationEnsemble = classificationEnsemble;

% Perform cross-validation
partitionedModel = crossval(Classifier.ClassificationEnsemble, 'KFold', K);

% Compute validation accuracy
Accuracy = 1 - kfoldLoss(partitionedModel, 'LossFun', 'ClassifError');

% Compute validation predictions and scores
[Predictions, Scores] = kfoldPredict(partitionedModel);
```

F.3 MAS - CRM analysis

F.3.1 CRM_data_extract.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function that extract information from the CRM analysis .txt files. The
% "Markers.txt" is used to calculate the LAG between MAS and CRM recording
% start times. Furthermore, the function creates a matrix
% "event_matrix.txt" containing information of each annotated event.
%
% INPUT
%   filepath:           - Folder where crm-data is located. The .txt files
%                       must must only contain time, not date.
%   OUTPUT
%   LAG                 - The difference (in samples) between MAS and CRM
%                       start
%
%   Author: Mathias P. Bonnesen, s113918, Technical University of Denmark.
%
%   Date: 11/21 - 2016
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [LAG] = CRM_data_extract(filepath)
current_folder = cd;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Find lag between CRM and MAS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%% Calculate LAG %%%%%%%%%
cd(filepath)

fileID = fopen('Markers.txt','r');        % read whole file
textfile = textscan(fileID,'%s','Delimiter','\n');
textfile = textfile{1};
time = textfile{2};

time_mo = str2num(time(16:17));
time_d = str2num(time(13:14));
time_h = str2num(time(24:25));
time_m = str2num(time(27:28));
time_s = str2num(time(30:31));

MAS_data = readtable([filepath(1:end-3) 'iFile.txt']);  % READ MAS time

time_moM = str2num(MAS_data{8,4}{1}(4:5));        % MAS start time (Month)
time_dM = str2num(MAS_data{8,4}{1}(1:2));         % MAS start time (Day)
time_hM = MAS_data{8,5};                          % MAS start time (Hour)
```

```
time_mM = str2num(MAS_data{9,1}{1}(2:3));          % MAS start time (Minute)
time_sM = str2num(MAS_data{9,1}{1}(5:6));          % MAS start time (Second)


date_check = (str2num(time(13:14))+str2num(time(16:17))*100) - ...
    (str2num(MAS_data{8,4}{1}(1:2)) + str2num(MAS_data{8,4}{1}(4:5))*100);

mo_diff = time_mo - time_mM;                              % Time difference
d_diff = time_d - time_dM;                               % Time difference
h_diff = time_h - time_hM;                               % Time difference
m_diff = time_m - time_mM;
s_diff = time_s - time_sM;

% Find TOTAL LAG in seconds
LAG = h_diff * 3600 + m_diff * 60 - s_diff...
    - 24*3600*(sign(m_diff)*sign(d_diff));   % Find MAS/CRM LAG (seconds)
Duration = 1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Find and store events %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%

event_matrix = cell(1,4);     % Create event matrix cell structure
count = 4;                    % Variable to keep track of event number
txtfiles = dir('*.txt');

    for i = 1:length(txtfiles)                % Loop through all .txt files
        filename = txtfiles(i).name;

        if isempty(strfind(filename,'Event')) == 0  % Neglect blank lines
            fileID = fopen(filename,'r');          % read whole file
            textfile = textscan(fileID,'%s','Delimiter','\n');
            textfile = textfile{:};
            for ii = 1:length(textfile)
               cellf = strsplit(textfile{ii},';');
               if length(cellf) > 1
                   time = cellf{1};          % Extract time

                   start = time(1:12);                % event start
                   stop = time(14:end);               % event stop
                   duration = cellf{2};               % duration
                   label = cellf{3};                  % event label

                  % Save them into a structure
                   event_matrix{count,1} = label;
                   event_matrix{count,2} = start;
                   event_matrix{count,3} = stop;
                   event_matrix{count,4} = duration;
                   count = count + 1;
               end
            end

            % When event .txt file is encountered, extract analysis start
            % time and lights ON
        elseif isempty(strfind(filename,'Markers')) == 0

            % START TIME
            fileID = fopen(filename,'r');          % read whole file
            textfile = textscan(fileID,'%s','Delimiter','\n');
            textfile = textfile{:};
            cellf = strsplit(textfile{2},';');
            timec = cellf{1};
            event_matrix{1,1} = timec(1:11);
            event_matrix{1,2} = timec(13:22);
            event_matrix{1,3} = timec(24:end);

            % END TIME
            textfile_stop = textfile{end};
            cellf = strsplit(textfile_stop,';');
            event_matrix{2,1} = textfile_stop(15:end);
            event_matrix{2,2} = textfile_stop(1:12);
        end
    end
```

```
    %%%% Write data into .txt file
        fileID2 = fopen('event_matrix.txt','w');
        formatSpec = '%s %s %s %s \n';
        [nrows,ncols] = size(event_matrix);
    for row = 1:nrows
        fprintf(fileID2,formatSpec,event_matrix{row,:});
    end
    fclose(fileID);
    cd(current_folder)
end
```

F.3.2 CRM_MAS_events.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This script compares the events found by the medical staff and the events
% located by MAS algorithm
%
% Author: Mathias P. Bonnesen, s113918, Technical University of Denmark.
%
% Date: 12/23 - 2016
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% LOAD and store events

current_folder = '\\10.230.149.201\MobilApnea\MATLAB'
filepath = ['\\10.230.149.201\MobilApnea\DATA\', Pat, '\crm'];

filename = [filepath,'\event_matrix.txt']

fileID = fopen(filename,'r');        % read whole file
textfile = textscan(fileID,'%s','Delimiter','\n');
textfile = textfile{:};
event_matrix = cell(1,3);     % Create event matrix cell structure
count = 1;

for ii = 1:length(textfile)
   cellf = strsplit(textfile{ii},';');
   cellf = cellf{:};
   if ii == 1
    starth_rec = str2num(cellf(24:25));        % event start, H,m,s
    startm_rec = str2num(cellf(27:28));
    starts_rec = str2num(cellf(30:31));


    add_sec = abs(starth_rec*3600+startm_rec*60+starts_rec - 24*3600);

   elseif ii > 3
       cellf = strsplit(textfile{ii},' ');
       time = cellf{end - 3};
       start_h = str2num(time(1:2));        % event start, H,m,s
       start_m = str2num(time(4:5));
       start_s = str2num(time(7:8));

          % See if the event is before midnight
        if start_h > 10;
            event_sample = add_sec + ((start_h+start_m/60+start_s/3600)-24)*3600;
        else
            event_sample = start_h*3600+start_m*60+start_s + add_sec;
        end

        cellf2 = strsplit(cellf{end-1},',');
       duration_s = [str2num(cellf2{1});str2num(cellf2{end})];
       duration = duration_s(1) + ceil(duration_s(end)/10); % event duration

       event_matrix{count,1} = int64(event_sample);              % start of event
       event_matrix{count,2} = int64(event_sample) + duration;   % end of event
       event_matrix{count,3} = cellf2{1};
       event_matrix{count,4} = cellf{1};
```

```
        count = count + 1;

    end

end
cd(current_folder)

%% PLOT Aud, Acc, Label and CRM results
window = 50;
crm_lab = zeros(1,floor(length(X_f)/fs_acc));
c = 0;

for i =300:size(event_matrix,1)
%for i = 702:size(event_matrix,1)
    if event_matrix{i,1} > -LAG ...
            && event_matrix{i,1}+LAG < length(X_f)/fs_acc

        start = event_matrix{i,1} + LAG;
        stop = event_matrix{i,2} + LAG;

        Inta = start*fs_aud-fs_aud*window:...
            stop*fs_aud + fs_aud*window;  % Look at events

        Intb = start*fs_acc-fs_acc*window:...
            stop*fs_acc + fs_acc*window;  % Look at events
%         m = max(audio_p(Int));
        [start stop];
        event_type = event_matrix{i,4}
         if strcmp(event_type,'Apnea') == 1;
             crm_lab(start:stop) = ones(length(start:stop),1)*(-1.5);
%               disp('bing')
                c = c + 1
%               pause
          elseif strcmp(event_type,'Hypopnea')
              %c = c + 1
              %crm_lab(start:stop) = ones(length(start:stop),1)*(-2);
          end

        aud_env = envelope(audio_p(Inta),fs_aud/2,'rms');
        figure(1)
        subplot(4,1,1)
        title(event_type)
        hold off
        plot(0:1/fs_aud:length(audio_p(Inta))/fs_aud-1/fs_aud,audio_p(Inta))
        title('Audio')
        hold on
        plot(0:1/fs_aud:length(audio_p(Inta))/fs_aud-1/fs_aud,lab_vec(Inta)*400);
        axis tight
        subplot(4,1,2)
        hold off
        plot(0:1/fs_acc:length(c_acc(Intb))/fs_acc-1/fs_acc,X_f(Intb));
        title('X Acc')
        axis tight
        subplot(4,1,3)
        hold off
        plot(0:1/fs_acc:length(c_acc(Intb))/fs_acc-1/fs_acc,Y_f(Intb));
        title('Y Acc')
        axis tight
        subplot(4,1,4)
        hold off
        plot(0:1/fs_acc:length(c_acc(Intb))/fs_acc-1/fs_acc,Z_f(Intb));
        title('X Acc')
        xlabel('Time (seconds)')
        axis tight
        figure(2)
        plot(rest_vec(1:end-3,start-window:stop+window)')
        %legend('RW_25','RW_5e','RW_18','RB_1','RB_2')
        axis tight
        legend('RW_25','RW_5e','RW_18')
        figure(3)
        hold off
        plot(pAvg_vec(start-window:stop+window))
        hold on
```

```
        plot(se_vec(start-window:stop+window))
        axis tight
        set(gca,'yscale','log','fontsize',18);
        legend('pAvg','Entropy')
        pause
    end
end
```

G Appendix: JAVA Code

This chapter contains all of the code used in the Android application. As stated in the program headers, the code is a modified version of previous master student Martin Guul.

G.1 Java application changes

This section lists all of the major changes made in the Android application since it was retrieved from Martin Guul.
0

Changes to be made	Problem	Plan	Fix	Notes	Progress
Crucial					
Only samples if its chosen to be set to high acc performance	Does not sample ACC data if not	Lock the Setting to high performance.	Settings-button disables, but not removed		
STOP-BANG	Remake the two first questions in order to	Only allow user to continue if ALL Q's are answered	Moved start-activity (SampleData) into an if-statement		
What if they accidentaly start the recording?	If they cancel recording and start new, no data will be saved	Change PatID each time a recording has been done	Changes it automaticly, to the current time	Can be improved by making the users unable to interact with the patient ID	
Start noise or artefact to syncronize MAS with CRM	This is very important if CRM data is to be used for supervised learning	The smartphone could make some noise when recording has begun	Use time-stamp vector instead.	Extracts tFile.txt instead of making noise, since this is the most secure way of syncronyzing	
BMI calculation	Can lead to confusion	Simplify. Make it clear where to click, or change	IF-loop looks for empty inputs.	1: It could be hard to find the calculator 2: An error occured when Hight and Weight	
Change Text to Danish?	Cant be sure that patients understands english	Change text	Made new file translated to danish.		
For Convinience					
New Patient	Can make for a small confusion	Make it inactive	Button inactive		
RR SIM	Irrelevant for patients	Disable it	Disabled		
Settings	Irrelevant for patients	Remove		In future versions, insert it only for doctores	
Screen Maneuvrering	Goes to the top each time a question is answered	Remove this feature, and let screen float		This however, could lead to the patients having trouble finding the next	
For Optimization					
Three sampling buttons; Acc, Audio, Acc and Audio	Just needs one button that sais START Recording			The app should be as simple as possible, and therefore only one	

G.2 MAS Android application code

This section contains all of the code; both .java and .xml, that the Android application for the signal acquisition consists of.

G.2.1 Java files

MainActivity.java

```
package master.testgraphview;

/**
 * Created by martinguul on 10/12/15.
 * Last Edited by Mathias Pinto Bonnesen, 15/10/16
 */

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

import android.app.Activity;
import android.app.ActivityManager;
import android.app.AlertDialog;
import android.app.Notification;
import android.app.NotificationManager;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.media.AudioManager;
import android.net.Uri;
import android.os.BatteryManager;
import android.os.Bundle;
import android.os.Environment;
import android.preference.PreferenceManager;
import android.support.v4.content.LocalBroadcastManager;
import android.util.Log;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import com.jjoe64.graphview.GraphView;
import com.jjoe64.graphview.Viewport;
import com.jjoe64.graphview.series.DataPoint;
import com.jjoe64.graphview.series.LineGraphSeries;

public class MainActivity extends Activity implements SensorEventListener {

    private LineGraphSeries<DataPoint> series_x;
    private LineGraphSeries<DataPoint> series_y;
    private LineGraphSeries<DataPoint> series_z;
    private TextView tvSampleNr;
    private Viewport viewport;
    private int lastX = 0;
    private int lastY = 0;
    private int lastZ = 0;
```

```
private boolean sample = false;
private boolean sampler = true;
private boolean audioSampler = true;
private int sample_number = 1;
// Accelerometer
private SensorManager senSensorManager;
private Sensor senAccelerometer, senOrientation;
private Button bNewSample, bOpenService, bAudio, bAccAudio;
private TextView tvOrientation;
private GraphView graph;
private DataPoint[] data_x, data_y, data_z;

private Intent intentAcc, intentAudio, intentAudioStatus, intentAccTimer;

private int measure  = 100;
private int counter = 0;
private long lastUpdate = 0;
private float last_x, last_y, last_z;
private float x = 0;
private float y = 0;
private float z = 0;

private int max_val = 10;
private int min_val = -10;

// Sampling
private FileWriter writer;
File direct;
private float sampleTimex, sampleTimey, sampleTimez;
private boolean dialogStatus = true;

private ImageView imgView;

// Position
private int broadcastCounter = 0;
private boolean init = false; // boolean to indicate if one winow is sampled
int statusPosition  = 9000;
private int w = 5; // window length
private float[][] acc_data = new float[w][3]; // init data array
private float[] tmp_array = new float[3]; // tmp data array for received acc data

// Audio
private AudioManager audioManager;

// notification
NotificationManager nmSampling;
Notification notifSampling;

// Preferences
SharedPreferences sharedPrefs;
SharedPreferences.Editor editor;

//Sample rerun due to error check
private int sampleCounter = 1;

//mic receiver listener
private micIntentReceiver micReceiver;
private IntentFilter micFilter;

//bat receiver listener
private battIntentReceiver batReceiver;

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

    bNewSample = (Button) findViewById(R.id.bNewSample);
    bOpenService = (Button) findViewById(R.id.bService);
    bAudio = (Button) findViewById(R.id.bAudio);
    bAccAudio = (Button)findViewById(R.id.bAccAudio);
```

```
        tvOrientation = (TextView)findViewById(R.id.tvOrientation);


        tvSampleNr = (TextView) findViewById(R.id.tvSampleNr);
        tvSampleNr.setText("Sample Nr.: " + sample_number);
        // we get graph view instance
        GraphView graph = (GraphView) findViewById(R.id.graph1);
        // data
        series_x = new LineGraphSeries<DataPoint>();
        series_y = new LineGraphSeries<DataPoint>();
        series_z = new LineGraphSeries<DataPoint>();

        graph.addSeries(series_x);
        graph.addSeries(series_y);
        graph.addSeries(series_z);

        // customize a little bit viewport
        viewport = graph.getViewport();
        viewport.setYAxisBoundsManual(true);
        viewport.setMinY(-10);
        viewport.setMaxY(10);
        viewport.setScrollable(true);
        viewport.setScalable(true);

        // Accelerometer
        senSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        senAccelerometer = senSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER↩
            );

        senSensorManager.registerListener(this, senAccelerometer, SensorManager.↩
            SENSOR_DELAY_FASTEST);

        intentAcc = new Intent(this, AccServiceThread.class);
        // TODO: 08/12/15 intent changed to test for audio class update
        intentAudio = new Intent(this, AudioRecorderService.class);
       // intentAudioStatus = new Intent(this,WiredInReceiver.class);
        intentAccTimer = new Intent(this, AccServiceTimer.class);


        Log.i("fs", senAccelerometer.getMinDelay() + ""); // output the min break ↩
            between samples in micro seconds.

        // LocalBroadcast for position
        LocalBroadcastManager.getInstance(this).registerReceiver(accReceiver, new ↩
            IntentFilter("Acc Sample"));

        // Audio
        audioManager = (AudioManager) getSystemService(Context.AUDIO_SERVICE);

        // LED notification
        nmSampling = ( NotificationManager ) getSystemService( NOTIFICATION_SERVICE );
        notifSampling = new Notification();

        // Preferences
        sharedPrefs = PreferenceManager.getDefaultSharedPreferences(this);

        PreferenceManager.setDefaultValues(this, R.xml.preferences, false); // ↩
            initiate default values

        // check for directory and save directory in sharedpreferences
        direct = new File(Environment.getExternalStoragePublicDirectory
                (Environment.DIRECTORY_DOWNLOADS), "Data_" + sharedPrefs.getString("↩
                    PatientName","test"));
        if (!direct.exists()) {
            direct.mkdirs();
        }

        editor = sharedPrefs.edit();
        editor.putString("directory", direct + "");
        editor.apply(); // apply in stead of commit since it runs on an other thread.

        // Memory information
```

```
        Log.i("Memory", ((ActivityManager) getSystemService(Context.ACTIVITY_SERVICE)↩
            ).getMemoryClass() + "");

        // mic state
        AudioManager au = (AudioManager) getSystemService(Context.AUDIO_SERVICE);
        micReceiver = new micIntentReceiver();
        micFilter = new IntentFilter(au.ACTION_HEADSET_PLUG);
        registerReceiver(micReceiver, micFilter);

        // Battery listener
        // TODO: 08/12/15 insert if loop to enable low battery detector
        /*
        batReceiver = new battIntentReceiver();
        IntentFilter battFilter = new IntentFilter(Intent.ACTION_BATTERY_LOW);
        registerReceiver(batReceiver, battFilter);*/
    }

    private class micIntentReceiver extends BroadcastReceiver{
        @Override
        public void onReceive(Context context, Intent intent) {
            int state = intent.getIntExtra("state", -2);
            int micState = intent.getIntExtra("microphone",-2);
            switch (state){
                case 0:
                    Toast.makeText(context, "No headset is mounted", Toast.↩
                        LENGTH_SHORT).show();
                    bAudio.setBackground(context.getResources().getDrawable(R.drawable↩
                        .button_red));
                    bAccAudio.setBackground(context.getResources().getDrawable(R.↩
                        drawable.button_red));
                    break;
                case 1:
                    if(micState == 0){
                        Toast.makeText(context, "Headset is mounted but no MIC", Toast↩
                            .LENGTH_SHORT).show();
                        bAudio.setBackground(context.getResources().getDrawable(R.↩
                            drawable.button_red));
                        bAccAudio.setBackground(context.getResources().getDrawable(R.↩
                            drawable.button_red));
                    }else if(micState == 1){
                    Toast.makeText(context, "MIC is mounted", Toast.LENGTH_SHORT).show↩
                        ();
                        bAudio.setBackground(context.getResources().getDrawable(R.↩
                            drawable.button_green));
                        bAccAudio.setBackground(context.getResources().getDrawable(R.↩
                            drawable.button_green));}
                    break;
                case -2:
                    Toast.makeText(context, "Wrong", Toast.LENGTH_SHORT).show();
                    break;
            }
        }
    }

    private class battIntentReceiver extends BroadcastReceiver{
        @Override
        public void onReceive(Context context, Intent intent) {
            // Power status
            IntentFilter ifilter = new IntentFilter(Intent.ACTION_BATTERY_CHANGED);
            Intent batteryStatus = context.registerReceiver(null, ifilter);

            float level = batteryStatus.getIntExtra(BatteryManager.EXTRA_LEVEL, -1);
            float scale = batteryStatus.getIntExtra(BatteryManager.EXTRA_SCALE, -1);

            float batteryPct = level / scale; // cast
            csv_generator("iFile", "Low Battery: " + batteryPct + " time: " + System.↩
                nanoTime());
        }
    }

    // Receive broadcast regarding position from AccServiceThread
    private BroadcastReceiver accReceiver = new BroadcastReceiver() {
```

```
        @Override
        public void onReceive(Context context, Intent intent) {
            int tmpStatus = intent.getIntExtra("acc_Position", 6);
            if(tmpStatus == 1){
                tvOrientation.setText("Position: Supine");
            } else if(tmpStatus == 2){
                tvOrientation.setText("Position: Prone");
            } else if(tmpStatus == 3){
                tvOrientation.setText("Position: Left Lateral");
            } else if(tmpStatus == 4){
                tvOrientation.setText("Position: Right Lateral");
            } else if(tmpStatus == -2){
                tvOrientation.setText("Position: Upright");
            } else if(tmpStatus == 5){
                tvOrientation.setText("Position: Nothing to Declare");
            } else if(tmpStatus == 6){
                tvOrientation.setText("Position: Can't Find Data");
            }
        }
    };

    public void unregisterReceivers(int state){
        if(state == 1){ // sampling data
            unregisterReceiver(micReceiver);}
        else if(state == 2){ // not sampling data
            registerReceiver(micReceiver, micFilter);
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        LocalBroadcastManager.getInstance(this).unregisterReceiver(accReceiver);
        LocalBroadcastManager.getInstance(this).unregisterReceiver(micReceiver);
        LocalBroadcastManager.getInstance(this).unregisterReceiver(batReceiver);
        //stopService(intentAudioStatus); // stop the listener for audio connection
        stopService(intentAudio); // stop sampling audio
        stopService(intentAcc); // stop sampling with accelerometer
        stopService(intentAccTimer);
    }

    public void sampling(View view){
        if (!sample){
            sample = true;
            sampleAcc();

            // defining directory for samples
            direct = new File(Environment.getExternalStoragePublicDirectory
                    (Environment.DIRECTORY_DOWNLOADS), "Data_" + sample_number);
            // set text on btn
            bNewSample.setText("Stop Recording");
            tvSampleNr.setText("Sample Nr.: " + sample_number);

            // inactivate the other buttons
            activateButtons(1);

            SamplingFlashLight();
        }
        else{
            sample = false;
            sampleAcc();

            // increase counter for number of sample epochs
            sample_number++;

            // set text on btn
            bNewSample.setText("UI Accelerometer");
            // cancel LED notification
            nmSampling.cancel(16);

            // activate buttons
            activateButtons(5);
        }
```

```
    }

    // notify that the device samples using LED.
    private void SamplingFlashLight() {
        notifSampling.ledARGB = 0xFFff0000; // choose color for LED
        notifSampling.flags = Notification.FLAG_SHOW_LIGHTS;
        notifSampling.ledOnMS = 100;
        notifSampling.ledOffMS = 100;
        nmSampling.notify(16, notifSampling);
    }

    protected void sampleAcc(){
        new Thread(new Runnable() {

            @Override
            public void run() {
                // we add 100 new entries
                while (sample) {
                    runOnUiThread(new Runnable() {

                        @Override
                        public void run() {
                            addEntry();

                        }
                    });
                    // sleep to slow down the add of entries
                    try {
                        Thread.sleep(100); // the delay time in milliseconds
                    } catch (InterruptedException e) {
                        // manage error
                    }
                }
            }
        }).start();
    }

    // add random data to graph
    private void addEntry() {
        // here, we choose to display max 10 points on the viewport and we scroll to ↩
            end
        series_x.appendData(new DataPoint(lastX++, x), true, 100);
        series_x.setColor(Color.BLUE);
        series_y.appendData(new DataPoint(lastY++, y), true, 100);
        series_y.setColor(Color.RED);
        series_z.appendData(new DataPoint(lastZ++, z), true, 100);
        series_z.setColor(Color.GREEN);

        csv_generator("x_coor.txt", x + "");
        csv_generator("timex.txt", sampleTimex + "");
        csv_generator("y_coor.txt", y + "");
        csv_generator("timey.txt", sampleTimey + "");
        csv_generator("z_coor.txt", z + "");
        csv_generator("timez.txt", sampleTimez + "");
    }

    @Override
    public void onSensorChanged(SensorEvent sensorEvent) {
        Sensor mySensor = sensorEvent.sensor;

        if (mySensor.getType() == Sensor.TYPE_ACCELEROMETER) {

            x = sensorEvent.values[0];
            sampleTimex = System.nanoTime();
            y = sensorEvent.values[1];
            sampleTimey = System.nanoTime();
            z = sensorEvent.values[2];
            sampleTimez = System.nanoTime(); // get time stamp

            // set the max and min value at view port
            if(max_val < x | max_val < y | max_val < z){
                max_val = (int) Math.max(x,(int) Math.max(y,z));
                viewport.setMaxY(max_val*1.3);
```

```
            }

            if(min_val < Math.abs(x) | min_val < Math.abs(y) | min_val < Math.abs(z)){
                min_val = (int) Math.min(x,(int) Math.min(y,z));
                viewport.setMinY(-10);
            }
        }
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {

    }

    public void csv_generator(String sFileName, String input){
        try
        {
            if (!direct.exists()) {
                direct.mkdirs();
            }
            File gpxfile = new File(direct, sFileName);
            FileWriter writer = new FileWriter(gpxfile, true);
            writer.append(input  + ", ");
            writer.flush();
            writer.close();
            updateMediaScanner(gpxfile);
        }
        catch(IOException e)
        {
            e.printStackTrace();
        }
    }

    public void updateMediaScanner(File file) {
        //Send an Intent to the MediaScanner to scan our file
        //Broadcast the Media Scanner Intent to trigger it
        Uri uri = Uri.fromFile(file);
        sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE, uri));
    }

    //-----------------------------------------------------------------
    //                  Acc data sampling using service
    //-----------------------------------------------------------------
    public void openService(View v) {
        if (sampler) {
            intentAcc.putExtra("samplingStatus",1); // number of modalities
            startService(intentAccTimer);
            bOpenService.setText("Stop Recording");
            sampler = false;
            // inactivate the other buttons
            activateButtons(2);

            // unregister non important receivers
            unregisterReceivers(1);
        }
        else{
            // TODO: 10/12/15 removed to save power.
            /*
            // LocalbroadcastManager to stop sampling and unregister listener on acc ↩
                thread
            Intent intentAccStatus = new Intent("Acc Sampler status");
            LocalBroadcastManager.getInstance(this).sendBroadcast(intentAccStatus);
*/
            stopService(intentAccTimer);
            bOpenService.setText("Accelerometer");
            sampler = true;
            // activate all buttons
            activateButtons(5);

            // unregister non important receivers
            unregisterReceivers(2);
        }
    }
```

```
//-------------------------------------------------------------------
//                  Audio sampling
//-------------------------------------------------------------------
public void audioSample(View v) {
   /* if(sampler && !audioManager.isWiredHeadsetOn()){
        dialogStatus = false;
        while(!dialogStatus){
            DialogInsertHeadset();
         // wait for the response form dialog
        }
    }*/

    if (sampler && dialogStatus) {
        intentAudio.putExtra("samplingStatus",1); // number of modalities
        startService(intentAudio); // start the audio sample service
        bAudio.setText("Stop Recording"); // change text on btn
        sampler = false; // update counter

        // inactivate the other buttons
        activateButtons(3);

        // unregister non important receivers
        unregisterReceivers(1);
    } else {
        stopService(intentAudio); // stop the audio sample service
        bAudio.setText("Audio"); // change text on btn
        sampler = true; // update counter
        // activate all buttons
        activateButtons(5);

        // register non important receivers
        unregisterReceivers(2);
    }
}

public void DialogInsertHeadset() {
    // create dialog
    AlertDialog.Builder alertDialogBuilder = new AlertDialog.
            Builder(this);
    // set message and title
    alertDialogBuilder.setMessage("Headset is not mounted. This will reduce the ↩
        quality of the recording");
    alertDialogBuilder.setTitle("Headset Error");

    // Create positive btn
    alertDialogBuilder.setPositiveButton("Continue",
            new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface arg0, int arg1) {
                    dialogStatus = true;
                }
            });
    // Create negative btn
    alertDialogBuilder.setNegativeButton("Cancel",
            new DialogInterface.OnClickListener() {

                @Override
                public void onClick(DialogInterface dialog,
                                    int which) {
                    sampler = true; // update counter
                }
            });
    // build the alertDialog
    alertDialogBuilder.show();
}


//-------------------------------------------------------------------
//                  Accelerometer and Audio sampling
//-------------------------------------------------------------------
public void accAudioSampling(View v) {
    /* if(!audioManager.isWiredHeadsetOn() && sampler) {
```

```
            DialogInsertHeadset();
        } else */
        if (sampler) {
            intentAudio.putExtra("samplingStatus", 2); // number of modalities, 2 ↩
                means no notification, is controlled by acc
            startService(intentAudio); // start the audio sample service
            intentAccTimer.putExtra("samplingStatus", 2); // number of modalities, 2 ↩
                => red LEDs
            startService(intentAccTimer); // start the accelerometer sample service
            bAccAudio.setText("Stop Recording"); // change text on btn
            sampler = false; // update counter
            // inactivate the other buttons
            activateButtons(4);

            // unregister non important receivers
            unregisterReceivers(1);
        } else {
            // LocalbroadcastManager to stop sampling and unregister listener on acc ↩
                thread
            Intent intentAccStatus = new Intent("Acc Sampler status");
            LocalBroadcastManager.getInstance(this).sendBroadcast(intentAccStatus);

            stopService(intentAudio); // stop the audio sample service
            stopService(intentAccTimer); // stop the accelerometer sample service
            bAccAudio.setText("Accelerometer & Audio"); // change text on btn
            sampler = true; // update counter
            // activate all buttons
            activateButtons(5);

            // unregister non important receivers
            unregisterReceivers(2);
        }
    }

    public void activateButtons(int input){
        switch(input){
            case 1:
                bAudio.setEnabled(false);
                bOpenService.setEnabled(false);
                bAccAudio.setEnabled(false);
                break;
            case 2:
                // inactivate all buttons
                bAudio.setEnabled(false);
                bAccAudio.setEnabled(false);
                bNewSample.setEnabled(false);
                break;
            case 3:
                // inactivate all buttons
                bOpenService.setEnabled(false);
                bAccAudio.setEnabled(false);
                bNewSample.setEnabled(false);
                break;
            case 4:
                // activate all buttons
                bAudio.setEnabled(false);
                bOpenService.setEnabled(false);
                bNewSample.setEnabled(false);
                break;
            case 5:
                // activate all buttons
                bAudio.setEnabled(true);
                bOpenService.setEnabled(true);
                bAccAudio.setEnabled(true);
                bNewSample.setEnabled(true);
                break;
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu items for use in the action bar
        MenuInflater inflater = getMenuInflater();
```

```
        inflater.inflate(R.menu.menu_main, menu);
        return super.onCreateOptionsMenu(menu);
    }


// REMOVE THE MENU ITEMS not Useable by the users
/*
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        Intent intent = null;

        switch(item.getItemId()){
            case R.id.action_settings:
                intent = new Intent(this, Settings.class);
                this.startActivity(intent);
                break;
            case R.id.action_add_patient:
                intent = new Intent(this, AddPatient.class);
                this.startActivity(intent);
                break;
            default:
                return super.onOptionsItemSelected(item);
        }

        return true;
    }
*/


}
```

AddPatient.java

```
package master.testgraphview;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.os.Environment;
import android.preference.PreferenceManager;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

/**
 * Created by martinguul on 02/11/15.
 * Last Edited by Mathias Pinto Bonnesen, 15/10/16
 */
public class AddPatient extends Activity {
```

```
    private Spinner spinnerSnoring, spinnerTired, spinnerObserved, spinnerPressure, ↩
        spinnerBMI, spinnerAge, spinnerNeck, spinnerGender;
    private static EditText PatientAge;
    private static EditText patientName;
    private SharedPreferences sharedPrefs;
    private SharedPreferences.Editor editor;

    private File iFile;
    //private EditText etW, etH;

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

        // preferences
        sharedPrefs = PreferenceManager.getDefaultSharedPreferences(this);
        editor = sharedPrefs.edit();

        // Spinners
        spinnerSnoring = (Spinner) findViewById(R.id.spinnerSnoring);
        spinnerTired = (Spinner) findViewById(R.id.spinnerTired);
        spinnerObserved = (Spinner) findViewById(R.id.spinnerObserved);
        spinnerPressure = (Spinner) findViewById(R.id.spinnerPressure);
        spinnerBMI = (Spinner) findViewById(R.id.spinnerBMI);
        //spinnerAge = (Spinner) findViewById(R.id.spinnerAge);
        spinnerNeck = (Spinner) findViewById(R.id.spinnerNeck);
        spinnerGender = (Spinner) findViewById(R.id.spinnerGender);

        patientName = (EditText) findViewById(R.id.etPatientName);
        patientName.setText(sharedPrefs.getString("etPatientName","NA"));

        PatientAge = (EditText) findViewById(R.id.etPatientAge);
        PatientAge.setText(sharedPrefs.getString("etPatientAge",""));


        // set up spinner array
        ArrayAdapter<CharSequence> STOP_bang_Answer = ArrayAdapter.createFromResource(↩
            this,
                R.array.STOP_bang, android.R.layout.simple_spinner_item);

        STOP_bang_Answer.setDropDownViewResource(android.R.layout.↩
            simple_spinner_dropdown_item);

        spinnerSnoring.setAdapter(STOP_bang_Answer);
        spinnerTired.setAdapter(STOP_bang_Answer);
        spinnerObserved.setAdapter(STOP_bang_Answer);
        spinnerPressure.setAdapter(STOP_bang_Answer);
        //spinnerAge.setAdapter(STOP_bang_Answer);
        spinnerNeck.setAdapter(STOP_bang_Answer);


        // Set up spinner for BMI answers
        ArrayAdapter<CharSequence> BMI_Answer = ArrayAdapter.createFromResource(this,
                R.array.BMI, android.R.layout.simple_spinner_item);

        BMI_Answer.setDropDownViewResource(android.R.layout.↩
            simple_spinner_dropdown_item);

        spinnerBMI.setAdapter(BMI_Answer);

        // Set up spinner for Gender answers
        ArrayAdapter<CharSequence> gender_Answer = ArrayAdapter.createFromResource(↩
            this,
                R.array.gender, android.R.layout.simple_spinner_item);

        gender_Answer.setDropDownViewResource(android.R.layout.↩
            simple_spinner_dropdown_item);


        spinnerGender.setAdapter(gender_Answer);

        patientName.setVisibility(View.GONE);
```

```
        patientName.setVisibility(View.GONE);

        //patientName.setText(sharedPrefs.getString("PatientName","NA"));
        Calendar c = Calendar.getInstance();
        SimpleDateFormat df = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss");
        String formattedDate = df.format(c.getTime());
        patientName.setText(formattedDate);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu items for use in the action bar
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menu_add_patient, menu);
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        String age = PatientAge.getText() + "";

        if(spinnerSnoring.getSelectedItemPosition() != 0
                && spinnerGender.getSelectedItemPosition() != 0 && spinnerBMI.↩
                    getSelectedItemPosition() != 0 &&
                spinnerNeck.getSelectedItemPosition() != 0 && spinnerPressure.↩
                    getSelectedItemPosition() != 0
                && spinnerObserved.getSelectedItemPosition() != 0 && spinnerTired.↩
                    getSelectedItemPosition() != 0
                && age != ""){

            Intent intent = null;
            switch(item.getItemId()){
                case R.id.action_fished:
                    String textInput = patientName.getText() + "";


                    // save patient name to preferences
                    // Store date in variable
                    editor.putString("PatientName", textInput); // id for patient
                    editor.putString("Quest_order: ", "STOP BANG"); // Order of ↩
                        questions
                    editor.putString("Quest",spinnerGender.getSelectedItemPosition()
                            + ", "  + age + ", " +
                            spinnerBMI.getSelectedItemPosition() + ", " +
                            spinnerSnoring.getSelectedItemPosition() + ", " +
                            spinnerTired.getSelectedItemPosition() + ", " +
                            spinnerObserved.getSelectedItemPosition() + ", " +
                            spinnerPressure.getSelectedItemPosition() + ", " +
                            spinnerNeck.getSelectedItemPosition());
                    editor.apply();

                    Calendar c = Calendar.getInstance();
                    SimpleDateFormat df = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss");
                    String formattedDate = df.format(c.getTime());
                    patientName = (EditText) findViewById(R.id.etPatientName);
                    patientName.setText(formattedDate);

                    // check and define directory
                    File direct = new File(Environment.↩
                        getExternalStoragePublicDirectory
                            (Environment.DIRECTORY_DOWNLOADS), "Data_" + sharedPrefs.↩
                                getString("PatientName","test"));

                    if (!direct.exists()) {
                        direct.mkdirs();
                        System.out.println("Checksome");
                    }
```

```
                    // START the SampleData.java class
                    intent = new Intent(this, SampleData.class);
                    this.startActivity(intent);
                    break;
            }

        } else{
            DialogMissingAnswer();
        }

        return true;
    }

    public void csv_info(File sFileName, String input) {
        try {
            FileWriter writer = new FileWriter(sFileName, true);
            writer.append(input + ", " + "\n");
            writer.flush();
            writer.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void DialogMissingAnswer() {
        // create dialog
        AlertDialog.Builder alertDialogBuilder = new AlertDialog.
                Builder(this);
        // set message and title
        alertDialogBuilder.setMessage("Svar venligst pÃ¥ alle spÃ¸rgsmÃ¥lene");
        alertDialogBuilder.setTitle("Fejl i besvarelse");

        // Create positive btn
        alertDialogBuilder.setPositiveButton("Ok",
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface arg0, int arg1) {

                    }
                });
        // Create negative btn
        alertDialogBuilder.setNegativeButton("Cancel",
                new DialogInterface.OnClickListener() {

                    @Override
                    public void onClick(DialogInterface dialog,
                                        int which) {

                    }
                });

        // build the alertDialog
        alertDialogBuilder.show();
    }

    public void DialogBMI(View view) {
        LayoutInflater factory = LayoutInflater.from(this);
        final View textEntryView = factory.inflate(R.layout.dialog_bmi, null);

        // create dialog
        final AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);

        alertDialogBuilder.setView(textEntryView);

        // set message and title
        alertDialogBuilder.setTitle("BMI Udregner");
        alertDialogBuilder.setMessage("IndsÃ¦t hÃ¸jde (cm) og vÃ¦gt (kg), og tryk OK")↩
            ;

        // Create positive btn
        alertDialogBuilder.setPositiveButton("Ok",
                new DialogInterface.OnClickListener() {
                    @Override
```

```
                    public void onClick(DialogInterface arg0, int arg1) {
                        EditText etW = (EditText) textEntryView.findViewById(R.id.etW)↩
                            ;
                        EditText etH = (EditText) textEntryView.findViewById(R.id.etH)↩
                            ;

                        // IF statement secures that app does not crash due to lag of ↩
                            inputs
                        if(etH.getText().length() > 0 && etW.getText().length() > 0) {
                            double BMI_val = Integer.valueOf(etW.getText() + "") /
                                    (Integer.valueOf(etH.getText() + "") *
                                            Integer.valueOf(etH.getText() + ""))↩
                                                /10000;

                            // STORE the variables globally for later use
                            SharedPreferences sp = getSharedPreferences("your_prefs", ↩
                                Activity.MODE_PRIVATE);
                            SharedPreferences.Editor editor = sp.edit();
                            editor.putInt("height", Integer.valueOf(etH.getText() + ↩
                                ""));
                            editor.putInt("weight", Integer.valueOf(etW.getText() + ↩
                                ""));
                            editor.commit();


                            if (BMI_val < 1) {
                                spinnerBMI.setSelection(1);
                            }
                        }
                    }
                });
        // Create negative btn
        alertDialogBuilder.setNegativeButton("Cancel",
                new DialogInterface.OnClickListener() {

                    @Override
                    public void onClick(DialogInterface dialog,
                                        int which) {

                    }
                });

        // build the alertDialog
        alertDialogBuilder.show();
    }


}
```

SampleData.java

```
package master.testgraphview;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.Notification;
import android.app.NotificationManager;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.SharedPreferences;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.net.Uri;
import android.os.BatteryManager;
import android.os.Bundle;
import android.os.Environment;
```

```
import android.preference.PreferenceManager;
import android.support.v4.content.LocalBroadcastManager;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;


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

/**
 * Created by martinguul on 10/12/15.
 * Last Edited by Mathias Pinto Bonnesen, 15/10/16
 */
public class SampleData extends Activity {

    private boolean sampler = true;

    private Button bNewSample, bOpenService, bAudio, bAccAudio;
    private TextView tvOrientation;

    private Intent intentAudio, intentAccTimer;

    // TEST SEGMENT
    private static EditText patientName;
    // TEST


    // notification
    NotificationManager nmSampling;
    Notification notifSampling;

    // Preferences
    SharedPreferences sharedPrefs;
    SharedPreferences.Editor editor;
    File direct;

    //mic receiver listener
    private micIntentReceiver micReceiver;
    private IntentFilter micFilter;

    //bat receiver listener
    private battIntentReceiver batReceiver;

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

        bNewSample = (Button) findViewById(R.id.bNewSample);
        bOpenService = (Button) findViewById(R.id.bService);
        bAudio = (Button) findViewById(R.id.bAudio);
        bAccAudio = (Button)findViewById(R.id.bAccAudio);
        tvOrientation = (TextView)findViewById(R.id.tvOrientation);

        intentAudio = new Intent(this, AudioRecorderService.class);
        intentAccTimer = new Intent(this, AccServiceTimer.class);

        // Sets the visibility of button 1 and 2 to zero
        bAudio.setVisibility(View.GONE);
        bOpenService.setVisibility(View.GONE);

        // TODO: 10/12/15 removed to save power
        /*
        // LocalBroadcast for position
```

```
        LocalBroadcastManager.getInstance(this).registerReceiver(accReceiver, new ↩
            IntentFilter("Acc Sample"));
*/

        // LED notification
        nmSampling = ( NotificationManager ) getSystemService( NOTIFICATION_SERVICE );
        notifSampling = new Notification();

        // Preferences
        sharedPrefs = PreferenceManager.getDefaultSharedPreferences(this);
        PreferenceManager.setDefaultValues(this, R.xml.preferences, false); // ↩
            initiate default values

        // check for directory and save directory in sharedpreferences
        direct = new File(Environment.getExternalStoragePublicDirectory
                (Environment.DIRECTORY_DOWNLOADS), "Data_" + sharedPrefs.getString("↩
                    PatientName","test"));
        if (!direct.exists()) {
            direct.mkdirs();
        }

        editor = sharedPrefs.edit();
        editor.putString("directory", direct + "");
        editor.apply(); // apply in stead of commit since it runs on an other thread.

        // mic state
        AudioManager au = (AudioManager) getSystemService(Context.AUDIO_SERVICE);
        micReceiver = new micIntentReceiver();
        micFilter = new IntentFilter(au.ACTION_HEADSET_PLUG);
        registerReceiver(micReceiver, micFilter);

        // Battery listener
        // TODO: 08/12/15 insert if loop to enable low battery detector
        /*
        batReceiver = new battIntentReceiver();
        IntentFilter battFilter = new IntentFilter(Intent.ACTION_BATTERY_LOW);
        registerReceiver(batReceiver, battFilter);*/

        System.out.println(sharedPrefs.getString("Quest", "NA"));
    }

    private class micIntentReceiver extends BroadcastReceiver {
        @Override
        public void onReceive(Context context, Intent intent) {
            int state = intent.getIntExtra("state", -2);
            int micState = intent.getIntExtra("microphone",-2);
            switch (state){
                case 0:
                    Toast.makeText(context, "No headset is mounted", Toast.↩
                        LENGTH_SHORT).show();
                    bAudio.setBackground(getDrawable(R.drawable.button_red));
                    bAccAudio.setBackground(getDrawable(R.drawable.button_red));
                    break;
                case 1:
                    if(micState == 0){
                        Toast.makeText(context, "Headset is mounted but no MIC", Toast↩
                            .LENGTH_SHORT).show();
                        bAudio.setBackground(getDrawable(R.drawable.button_red));
                        bAccAudio.setBackground(getDrawable(R.drawable.button_red));
                    }else if(micState == 1){
                        Toast.makeText(context, "MIC is mounted", Toast.LENGTH_SHORT).↩
                            show();
                        bAudio.setBackground(getDrawable(R.drawable.button_green));
                        bAccAudio.setBackground(getDrawable(R.drawable.button_green))↩
                            ;}
                    break;
                case -2:
                    Toast.makeText(context, "Wrong", Toast.LENGTH_SHORT).show();
                    break;
            }
        }
    }
```

```
    // TODO: 10/12/15 not used since saving power
    private class battIntentReceiver extends BroadcastReceiver{
        @Override
        public void onReceive(Context context, Intent intent) {
            // Power status
            IntentFilter ifilter = new IntentFilter(Intent.ACTION_BATTERY_CHANGED);
            Intent batteryStatus = context.registerReceiver(null, ifilter);

            float level = batteryStatus.getIntExtra(BatteryManager.EXTRA_LEVEL, -1);
            float scale = batteryStatus.getIntExtra(BatteryManager.EXTRA_SCALE, -1);

            float batteryPct = level / scale; // cast
            csv_generator("iFile", "Low Battery: " + batteryPct + " time: " + System.↩
                nanoTime());
        }
    }

    // TODO: 10/12/15 not used since saving power
    // Receive broadcast regarding position from AccServiceThread
    private BroadcastReceiver accReceiver = new BroadcastReceiver() {

        @Override
        public void onReceive(Context context, Intent intent) {
            int tmpStatus = intent.getIntExtra("acc_Position", 6);
            if(tmpStatus == 1){
                tvOrientation.setText("Position: Supine");
            } else if(tmpStatus == 2){
                tvOrientation.setText("Position: Prone");
            } else if(tmpStatus == 3){
                tvOrientation.setText("Position: Left Lateral");
            } else if(tmpStatus == 4){
                tvOrientation.setText("Position: Right Lateral");
            } else if(tmpStatus == -2){
                tvOrientation.setText("Position: Upright");
            } else if(tmpStatus == 5){
                tvOrientation.setText("Position: Nothing to Declare");
            } else if(tmpStatus == 6){
                tvOrientation.setText("Position: Can't Find Data");
            }
        }
    };

    public void csv_generator(String sFileName, String input){
        try
        {
            File gpxfile = new File(direct, sFileName);
            FileWriter writer = new FileWriter(gpxfile, true);
            writer.append(input  + ", ");
            writer.flush();
            writer.close();
            updateMediaScanner(gpxfile);
        }
        catch(IOException e)
        {
            e.printStackTrace();
        }
    }

    public void updateMediaScanner(File file) {
        //Send an Intent to the MediaScanner to scan our file
        //Broadcast the Media Scanner Intent to trigger it
        Uri uri = Uri.fromFile(file);
        sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE, uri));
    }

    public void unregisterReceivers(int state){
        if(state == 1){ // sampling data
            unregisterReceiver(micReceiver);}
        else if(state == 2){ // not sampling data
            registerReceiver(micReceiver, micFilter);
        }
    }
```

```
    @Override
    protected void onDestroy() {
        super.onDestroy();
        LocalBroadcastManager.getInstance(this).unregisterReceiver(accReceiver);
        LocalBroadcastManager.getInstance(this).unregisterReceiver(micReceiver);
        LocalBroadcastManager.getInstance(this).unregisterReceiver(batReceiver);
        //stopService(intentAudioStatus); // stop the listener for audio connection
        stopService(intentAudio); // stop sampling audio
        stopService(intentAccTimer);
    }

    // notify that the device samples using LED.
    private void SamplingFlashLight() {
        notifSampling.ledARGB = 0xFFff0000; // choose color for LED
        notifSampling.flags = Notification.FLAG_SHOW_LIGHTS;
        notifSampling.ledOnMS = 100;
        notifSampling.ledOffMS = 100;
        nmSampling.notify(16, notifSampling);
    }

    //-------------------------------------------------------------------
    //                  Acc data sampling using service
    //-------------------------------------------------------------------
    public void openService(View v) {
        if (sampler) {

            intentAccTimer.putExtra("samplingStatus",1); // number of modalities
            startService(intentAccTimer);
            bOpenService.setText("Stop Recording");
            sampler = false;
            // inactivate the other buttons
            activateButtons(2);

            // unregister non important receivers
            unregisterReceivers(1);

        }
        else{
            // TODO: 10/12/15 removed to save power.
            /*
            // LocalbroadcastManager to stop sampling and unregister listener on acc ↩
                thread
            Intent intentAccStatus = new Intent("Acc Sampler status");
            LocalBroadcastManager.getInstance(this).sendBroadcast(intentAccStatus);
*/
            stopService(intentAccTimer);
            bOpenService.setText("Accelerometer");
            sampler = true;
            // activate all buttons
            activateButtons(5);

            // unregister non important receivers
            unregisterReceivers(2);
        }
    }

    //-------------------------------------------------------------------
    //                  Audio sampling
    //-------------------------------------------------------------------
    public void audioSample(View v) {
       /* if(sampler && !audioManager.isWiredHeadsetOn()){
            dialogStatus = false;
            while(!dialogStatus){
                DialogInsertHeadset();
             // wait for the response form dialog
            }
        }*/

        if (sampler) {
            intentAudio.putExtra("samplingStatus",1); // number of modalities

            startService(intentAudio); // start the audio sample service
            bAudio.setText("Stop Recording"); // change text on btn
```

```
            sampler = false; // update counter

            // inactivate the other buttons
            activateButtons(3);

            // unregister non important receivers
            unregisterReceivers(1);

        } else {
            stopService(intentAudio); // stop the audio sample service
            bAudio.setText("Audio"); // change text on btn
            sampler = true; // update counter
            // activate all buttons
            activateButtons(5);

            // register non important receivers
            unregisterReceivers(2);
        }
    }

    public void DialogInsertHeadset() {
        // create dialog
        AlertDialog.Builder alertDialogBuilder = new AlertDialog.
                Builder(this);
        // set message and title
        alertDialogBuilder.setMessage("Headset is not mounted. This will reduce the ↩
            quality of the recording");
        alertDialogBuilder.setTitle("Headset Error");

        // Create positive btn
        alertDialogBuilder.setPositiveButton("Continue",
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface arg0, int arg1) {
                        //dialogStatus = true;
                    }
                });
        // Create negative btn
        alertDialogBuilder.setNegativeButton("Cancel",
                new DialogInterface.OnClickListener() {

                    @Override
                    public void onClick(DialogInterface dialog,
                                        int which) {
                        sampler = true; // update counter
                    }
                });
        // build the alertDialog
        alertDialogBuilder.show();
    }


    //-----------------------------------------------------------------
    //                  Accelerometer and Audio sampling
    //-----------------------------------------------------------------
    public void accAudioSampling(View v) {
        /* if(!audioManager.isWiredHeadsetOn() && sampler) {
            DialogInsertHeadset();
        } else */
        if (sampler) {

            // TEST SEGMENT
            /*
            Calendar c = Calendar.getInstance();
            SimpleDateFormat df = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss");
            String formattedDate = df.format(c.getTime());

            editor.putString("PatientName", formattedDate); // id for patient
            editor.apply();
            !! TEST SEGMENT */

            /*  NOISE SEGMENT, UNCOMMENDED FOR DEVELOPMENT-REASONS
            // SET VOLUME TO MAX and force to use SPEAKERS
```

```
            AudioManager am = (AudioManager) getSystemService
                    (Context.AUDIO_SERVICE);
            am.setStreamVolume(AudioManager.STREAM_MUSIC,
                    am.getStreamMaxVolume(AudioManager.STREAM_MUSIC),0);
            am.setSpeakerphoneOn(true);
            am.setMode(AudioManager.MODE_IN_COMMUNICATION);

            // PLAY AUDIO, TO SYNCRONIZE MAS with the PSG
            final MediaPlayer mp = MediaPlayer.create(this, R.raw.test2);
            mp.start();
            */

            intentAudio.putExtra("samplingStatus", 2); // number of modalities, 2 ↩
                means no notification, is controlled by acc
            startService(intentAudio); // start the audio sample service
            intentAccTimer.putExtra("samplingStatus", 3); // number of modalities, 2 ↩
                => red LEDs
            startService(intentAccTimer); // start the accelerometer sample service
            bAccAudio.setText("Stop"); // change text on btn
            sampler = false; // update counter
            // inactivate the other buttons
            activateButtons(4);

            // unregister non important receivers
            unregisterReceivers(1);
        } else {
            // LocalbroadcastManager to stop sampling and unregister listener on acc ↩
                thread
            Intent intentAccStatus = new Intent("Acc Sampler status");
            LocalBroadcastManager.getInstance(this).sendBroadcast(intentAccStatus);

            stopService(intentAudio); // stop the audio sample service
            stopService(intentAccTimer); // stop the accelerometer sample service
            bAccAudio.setText("Start Dataopsamling"); // change text on btn
            sampler = true; // update counter
            // activate all buttons
            activateButtons(5);


            // Go back to previous activity. This was implemented, as a fix
            this.finish();

            // unregister non important receivers
            unregisterReceivers(2);
        }
    }

    public void activateButtons(int input){
        switch(input){
            case 1:
                bAudio.setEnabled(false);
                bOpenService.setEnabled(false);
                bAccAudio.setEnabled(false);
                break;
            case 2:
                // inactivate all buttons
                bAudio.setEnabled(false);
                bAccAudio.setEnabled(false);
                break;
            case 3:
                // inactivate all buttons
                bOpenService.setEnabled(false);
                bAccAudio.setEnabled(false);
                break;
            case 4:
                // activate all buttons
                bAudio.setEnabled(false);
                bOpenService.setEnabled(false);
                break;
            case 5:
                // activate all buttons
                bAudio.setEnabled(true);
                bOpenService.setEnabled(true);
```

```
                bAccAudio.setEnabled(true);
                break;
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu items for use in the action bar
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menu_sampledata, menu);
        return super.onCreateOptionsMenu(menu);
    }

    /* THIS HAS BEEN REMOVED, FOR USER SIMPLICITY

    @Override

    public boolean onOptionsItemSelected(MenuItem item) {
        Intent intent = null;

        switch(item.getItemId()){
            case R.id.action_settings:
                intent = new Intent(this, Settings.class);
                this.startActivity(intent);
                break;
            case R.id.action_add_patient:
                intent = new Intent(this, AddPatient.class);
                this.startActivity(intent);
                break;
            default:
                return super.onOptionsItemSelected(item);
        }

        return true;
    }
    */


}
```

AudioRecorderService.java

```
package master.testgraphview;

import android.app.Service;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.SharedPreferences;
import android.media.AudioFormat;
import android.media.AudioRecord;
import android.media.MediaRecorder;
import android.net.Uri;
import android.os.BatteryManager;
import android.os.Environment;
import android.os.IBinder;
import android.preference.PreferenceManager;
import android.support.annotation.Nullable;
import android.util.Log;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Calendar;

/**
 * Created by martinguul on 01/12/15.
```

```
 * inspiration: http://stackoverflow.com/questions/8499042/android-audiorecord-example↩
     /13487250#13487250
 * Last Edited by Mathias Pinto Bonnesen, 15/10/16
 */
public class AudioRecorderService extends Service {

    private int RECORDER_SAMPLERATE; // static final int RECORDER_SAMPLERATE = 44100;
    private static final int RECORDER_CHANNELS = AudioFormat.CHANNEL_IN_MONO;
    private static final int RECORDER_AUDIO_ENCODING = AudioFormat.ENCODING_PCM_16BIT;
    private int BufferElements2Rec;
    private static final int BytesPerElement = 2;
    private AudioRecord recorder = null;
    private Thread recordingThread = null;
    private boolean isRecording = false;
    private String audioDirectString;
    private SharedPreferences sharedPrefs;

    // file system
    private File iFile;

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

        //Preferences
        sharedPrefs = PreferenceManager.getDefaultSharedPreferences(this);

        //init values from preferences
        RECORDER_SAMPLERATE = Integer.valueOf(sharedPrefs.getString("AudioFs", ↩
            "16000")); // [Hz]
        BufferElements2Rec = Integer.valueOf(sharedPrefs.getString("AudioPackage", ↩
            "1024"));

        int bufferSize = AudioRecord.getMinBufferSize(RECORDER_SAMPLERATE,
                RECORDER_CHANNELS, RECORDER_AUDIO_ENCODING);


        // String defining directory
        audioDirectString = Environment.getExternalStoragePublicDirectory
                (Environment.DIRECTORY_DOWNLOADS) + "/Data_" +
                sharedPrefs.getString("PatientName","test");

        startRecording();

        // file system
        //iFile = new File(audioDirectString, "iFile.txt");


        // batteryStatus(1);
    }

    @Override
    public void onDestroy() {
        stopRecording();

        //batteryStatus(2);
        //updateMediaScanner(iFile);
        super.onDestroy();
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    private void startRecording() {
        recorder = new AudioRecord(MediaRecorder.AudioSource.MIC,
                RECORDER_SAMPLERATE, RECORDER_CHANNELS,
                RECORDER_AUDIO_ENCODING, BufferElements2Rec * BytesPerElement);
        // AudioRecord(int audioSource, int sampleRateInHz, int channelConfig,
        // int audioFormat, int bufferSizeInBytes)
```

```
        recorder.startRecording();
        isRecording = true;
        recordingThread = new Thread(new Runnable() {
            public void run() {
                writeAudioDataToFile();
            }
        }, "AudioRecorder Thread"); //Thread(runnable, thread name);
        //csv_info(iFile, "Fs aud: " + RECORDER_SAMPLERATE);
        //csv_info(iFile, "Package aud: " + BufferElements2Rec);

        recordingThread.start();
    }

    //convert short to byte
    private byte[] short2byte(short[] sData) {
        int shortArrsize = sData.length;
        byte[] bytes = new byte[shortArrsize * 2];
        for (int i = 0; i < shortArrsize; i++) {
            bytes[i * 2] = (byte) (sData[i] & 0x00FF);
            bytes[(i * 2) + 1] = (byte) (sData[i] >> 8);
            sData[i] = 0;
        }
        return bytes;

    }

    private void writeAudioDataToFile() {
        // Write the output audio in byte
        short sData[] = new short[BufferElements2Rec];

        FileOutputStream os = null;
        try {
            os = new FileOutputStream(audioDirectString + "/audio.pcm");
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }

        while (isRecording) {
            // gets the voice output from microphone to byte format
            recorder.read(sData, 0, BufferElements2Rec);
            //System.out.println("Short writing to file" + sData.toString());
            try {
                // // writes the data to file from buffer
                // // stores the voice buffer
                byte bData[] = short2byte(sData);
                os.write(bData, 0, BufferElements2Rec * BytesPerElement);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        try {
            os.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void stopRecording() {
        // stops the recording activity
        if (null != recorder) {
            isRecording = false;
            recorder.stop();
            recorder.release();
            recorder = null;
            recordingThread = null;

            // update media scanner
            File tmp = new File(audioDirectString + "/audio.pcm");
            Uri uri = Uri.fromFile(tmp);
            sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE, uri));
        }
    }
```

```
    public void batteryStatus(int status){
        // Power status
        IntentFilter ifilter = new IntentFilter(Intent.ACTION_BATTERY_CHANGED);
        Intent batteryStatus = this.registerReceiver(null, ifilter);

        float level = batteryStatus.getIntExtra(BatteryManager.EXTRA_LEVEL, -1);
        float scale = batteryStatus.getIntExtra(BatteryManager.EXTRA_SCALE, -1);

        float batteryPct = level / scale; // cast

        switch (status){
            case 1:
                //csv_info(iFile, "startBatteryStatus: " + batteryPct);
                //csv_info(iFile, "Fs aud: " + RECORDER_SAMPLERATE);
                break;
            case 2:
                //csv_info(iFile, "endBatteryStatus: " + batteryPct);
                break;
        }

    }

    public void csv_info(File sFileName, String input) {
        try {
            FileWriter writer = new FileWriter(sFileName, true);
            writer.append(input + ", " + "\n");
            writer.flush();
            writer.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void updateMediaScanner(File file) {
        //Send an Intent to the MediaScanner to scan our file
        //Broadcast the Media Scanner Intent to trigger it
        Uri uri = Uri.fromFile(file);
        sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE, uri));
    }
}
```

AccServiceTimer.java

```
package master.testgraphview;

import android.app.Activity;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.Service;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.SharedPreferences;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.net.Uri;
import android.os.BatteryManager;
import android.os.Environment;
import android.os.IBinder;
import android.preference.PreferenceManager;
import android.support.annotation.Nullable;
import android.support.v4.content.LocalBroadcastManager;
import android.util.Log;

import java.io.File;
import java.io.FileWriter;
```

```
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Calendar;

/**
 * Created by martinguul on 08/10/15.
 * Last Edited by Mathias Pinto Bonnesen, 15/10/16
 */
public class AccServiceTimer extends Service {

    private static int acc_T;
    private static int package_size;
    private String xBuf, yBuf, zBuf, tBuf; // string to collect sample time, and ↩
        buffers for writer
    private boolean sample = true; // boolean to control sampling on/off
    private  boolean sensorActive = false;
    private SensorManager senSensorManager;
    private Sensor senAccelerometer;
    private boolean sampleStarted = false;
    int stopsamp = 0;        // To control if the accelerometer measurements gets ↩
        written into the
                             // text-file.
    private Context context;

    private int counter = 1; // counter to build packets
    private File direct; // defining directory for samples

    // TODO: 08/12/15 changed to be called in the function to avoid memory drift
    //private float[] accData; // array for acc samples

    private Thread t_sample, t_write; // initiate thread for sampling

    private File file, xFile, yFile, zFile, tFile, pFile, iFile, eFile;

    // TODO: 09/12/15 position has been removed to lower power consumption
    /*
    // Position
    private int broadcastCounter = 0;
    private boolean init = false; // boolean to indicate if one window is sampled
    int statusPosition  = 9000;
    private int w = 5; // window length
    private float[][] acc_data = new float[w][3]; // init data array
    */

    // notification
    private NotificationManager nmSampling;
    private Notification notifSampling;
    private int samplingType;
    private boolean activatedLED = false;
    private boolean bfailSampling = true;

    //Preferences
    private SharedPreferences sharedPrefs;

    //batterystatus
    float batteryPct;

    @Override
    public void onStart(Intent intent, int startId) {
        super.onStart(intent, startId);
        samplingType = intent.getIntExtra("samplingStatus",1); // passed via intent ↩
            used to start service
    }

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

        // Preferences
        sharedPrefs = PreferenceManager.getDefaultSharedPreferences(this);

        context = this;
        //accData = new float[3]; // array for acc sampling.
```

```
        // check and define directory

        direct = new File(Environment.getExternalStoragePublicDirectory
                (Environment.DIRECTORY_DOWNLOADS), "Data_"
                + sharedPrefs.getString("PatientName","test"));

        if (direct.exists()) {

        } else {
            direct.mkdirs();

        }

        // LED notification
        nmSampling = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
        notifSampling = new Notification();

        // sampling settings
        acc_T = 1000000000/Integer.valueOf(sharedPrefs.getString("AccFs", "NA")); //[↩
            ns]
        package_size = Integer.valueOf(sharedPrefs.getString("AccPackage", "50")); //[↩
            samples]

        // File directories/paths
        xFile = new File(direct, "xFile.txt");
        yFile = new File(direct, "yFile.txt");
        zFile = new File(direct, "zFile.txt");
        tFile = new File(direct, "tFile.txt");
        pFile = new File(direct, "pFile.txt");
        iFile = new File(direct, "iFile.txt");
        eFile = new File(direct, "eFile.txt");


        SharedPreferences sp = getSharedPreferences("your_prefs", Activity.↩
            MODE_PRIVATE);
        int height = sp.getInt("height", -1);
        int weight = sp.getInt("weight", -1);
        csv_info(iFile,"hight: " + height + " weight: " + weight);


        // This is where the iFile and eFile is created
        csv_info(iFile, "Delta t [ms]: " + acc_T);
        csv_info(iFile, "Start Time [ns]: " + System.nanoTime());
        csv_info(iFile, "Package Size [samples]: " + package_size);
        // csv_info(iFile, "Audio Bit Rate [bits/sample]: " + sharedPrefs.getString("↩
            AudioBitRate", "NA"));
        // csv_info(iFile, "Audio fs [Hz]: " + sharedPrefs.getString("AudioFs", "NA"))↩
            ;
        // csv_info(iFile, "Stop Bang Answers: " + sharedPrefs.getString("Quest", "NA↩
            "));
        csv_info(iFile, "Stop Bang Answers: " + sharedPrefs.getString("Quest", "NA"));
        csv_info(eFile, "init string, ");

        // Get battery status from start
        batteryStatus(1);


        // start thread for sampling
        t_sample=new Thread(new AccSampler(), "Accelerometer Sample Thread"); // ↩
            sample on a new thread.
        t_sample.start();
    }

    @Override
    public void onDestroy() { // when service is destroyed
        sample = false; // stop sampling

        // cancel LED notification
        nmSampling.cancel(17);

        // Get end battery status
        batteryStatus(2);
```

```
        // turn off sample thread
        t_sample.interrupt();
        // update mediascanners
        updateMediaScanner(xFile);
        updateMediaScanner(yFile);
        updateMediaScanner(zFile);
        updateMediaScanner(tFile);
        updateMediaScanner(pFile);
        updateMediaScanner(iFile);
        updateMediaScanner(eFile);
        super.onDestroy();
    } // WORKS!



    public void csv_info(File sFileName, String input) {
        try {
            FileWriter writer = new FileWriter(sFileName, true);
            writer.append(input + ", " + "\n");
            writer.flush();
            writer.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void batteryStatus(int status){
        // Power status
        IntentFilter ifilter = new IntentFilter(Intent.ACTION_BATTERY_CHANGED);
        Intent batteryStatus = this.registerReceiver(null, ifilter);

        float level = batteryStatus.getIntExtra(BatteryManager.EXTRA_LEVEL, -1);
        float scale = batteryStatus.getIntExtra(BatteryManager.EXTRA_SCALE, -1);

        batteryPct = level / scale; // cast

        switch (status){
            case 1:
                Calendar c = Calendar.getInstance();
                SimpleDateFormat df = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss");
                String formattedDate = df.format(c.getTime());
                csv_info(iFile, "Date and time: " + formattedDate );
                csv_info(iFile, "startBatteryStatus: " + batteryPct);
                stopsamp = 0; // Write down the accelerometer data.
                break;
            case 2:
                Calendar r = Calendar.getInstance();
                SimpleDateFormat dr = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss");
                String formattedDateEnd = dr.format(r.getTime());
                csv_info(iFile, "Date and time: " + formattedDateEnd );
                csv_info(iFile, "endBatteryStatus: " + batteryPct);
                stopsamp = 1; // Stop recording the accelerometer data.
                break;
        }
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {

        return null;
    }

    public void updateMediaScanner(File file) {
        //Send an Intent to the MediaScanner to scan our file
        //Broadcast the Media Scanner Intent to trigger it
        Uri uri = Uri.fromFile(file);
        sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE, uri));
    } // WORKS!

    //-------------------------------------------------------------------------------------------↩
```

```
    // class for acc sampling, runs on a separate thread.
    //-------------------------------------------------------------------------------------------------------------------

    public class AccSampler implements SensorEventListener, Runnable {
        long tStamp, tStampUpdate; // currentTime, nextSampleTime
        double timeDif; // max diff time
        boolean firstSample = true;
        boolean sampleType = false; // normal sample ie game delay;

        @Override
        public void run() {
            // settings for accelerometer.
            senSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
            senAccelerometer = senSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
            if(sharedPrefs.getBoolean("AccPerformance", false) == false){ // normal performance
                senSensorManager.registerListener(this, senAccelerometer, SensorManager.SENSOR_DELAY_GAME, SensorManager.SENSOR_STATUS_ACCURACY_HIGH);
            } else if(sharedPrefs.getBoolean("AccPerformance", false) == true){ // high performance, may reduce power!!
                senSensorManager.registerListener(this, senAccelerometer, SensorManager.SENSOR_DELAY_FASTEST, SensorManager.SENSOR_STATUS_ACCURACY_HIGH);
                sampleType = false;
            }

            timeDif = acc_T * 0.5; //50 % of difference in the sample period in [ns] is accepted
            System.out.println(sharedPrefs.getBoolean("AccPerformance", false) + "");
            //-------------------------------------------------------------------------------------------------------------------

            // Check that sampling will not begin before the sensor is activated to avoid 0 as first
            // measure. Boolean is activated onSensorChange detection.
            //-------------------------------------------------------------------------------------------------------------------

            while(!sensorActive){
                try {
                    Thread.sleep(1);

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

            // TODO: 10/12/15 removed to save power
            /*
            // LocalBroadcast for position
            LocalBroadcastManager.getInstance(context).registerReceiver(accSample, new IntentFilter("Acc Sampler status"));
            */

            //-------------------------------------------------------------------------------------------------------------------

            // Start sampling until push button again on UI.
            //-------------------------------------------------------------------------------------------------------------------

            while(sample=true){ // no ideal solution!!
                //
            }
            if(!sample) { // sample updates by localbroadcast receiver.
                senSensorManager.unregisterListener(this); // unregister listener to reduce power consumption
                // cancel LED notification
                nmSampling.cancel(17);

            }
        }

        // TODO: 10/12/15 removed to save power
```

```
        /*
        // Receive broadcast regarding stop sampling from main activity
        private BroadcastReceiver accSample = new BroadcastReceiver() {
            @Override
            public void onReceive(Context context, Intent intent) {
                sample = false; // stop sampling and unregister acc listener on ↩
                    sampling thread.
            }
        };*/
        @Override
        public void onSensorChanged(SensorEvent event) {
            if(sampleType){ // Sampling method depends on the settings. This is for ↩
                game delay

                Sensor mySensor = event.sensor; // create sensor.
                sensorActive = true; // boolean to hold thread in the beginning until ↩
                    sensor is activated  to avoid initial zeros
                if (mySensor.getType() == Sensor.TYPE_ACCELEROMETER) {
                    //System.out.println(event.timestamp);
                    //System.out.println(tStampUpdate);
                    //System.out.println(tStampUpdate + timeDif);
                    //System.out.println(firstSample);


                    if(event.timestamp >= tStampUpdate && event.timestamp  <= (↩
                        tStampUpdate + timeDif)|| firstSample) { // Only sample if ↩
                        matches Fs or first sample
                        float[] accData = event.values; // acquire data x, y, and z ↩
                            from sensor. // TODO: 08/12/15 check if it is okay that ↩
                            timestamp and evnt dat a not are together?!
                        addEntry(accData);
                        System.out.println("I Made it !");

                        if (counter < package_size){ // save samples in buffers, less ↩
                            than package_size since else includes the last sample
                            if(counter > 1) { // check for first sample
                                xBuf = xBuf + event.values[0] + ", ";
                                yBuf = yBuf + event.values[1] + ", ";
                                zBuf = zBuf + event.values[2] + ", ";
                                tBuf = tBuf + event.timestamp  + ", ";
                            }else{ // first sample
                                xBuf = event.values[0] + ", ";
                                yBuf = event.values[1] + ", ";
                                zBuf = event.values[2] + ", ";
                                tBuf = event.timestamp  + ", ";
                            }
                            tStampUpdate = event.timestamp + acc_T; // The time for ↩
                                next sample [ns]
                            System.out.println(counter);
                            counter++; // update counter
                        }else{
                            counter = 1;
                            csv_generator(1, xBuf + event.values[0]);
                            csv_generator(2, yBuf + event.values[1]);
                            csv_generator(3, zBuf + event.values[2]);
                            csv_generator(4, tBuf + event.timestamp);
                            xBuf = yBuf = zBuf = tBuf = null;
                        }
                        // TODO: 09/12/15 position has been removed to lower power ↩
                            consumption
                        // Compute position
                        //position();
                        //System.out.println("Data sampled");
                        //tStampUpdate = event.timestamp + acc_T; // The time for next↩
                             sample [ns]
                        firstSample = false; // boolean for first sample to access if ↩
                            statement.
                    }else if(event.timestamp  > tStampUpdate + acc_T * 2) {
                        // TODO: 08/12/15 check what happens!
                        Log.i("Sample Error", "Did not sample");
                        csv_generator(6, "Sample Error at [ns]: " + event.timestamp); ↩
                            // log error to file.
                        // TODO: 10/12/15 removed to save power
```

```
                    nmSampling.cancel(17); // cancel notification to stop LED
                }
                    // update to new ideal time
                    //tStampUpdate = event.timestamp + acc_T;
                    // cancel LED notification
                    if(bfailSampling){
                        nmSampling.cancel(17);
                        samplingType = 2;
                        bfailSampling = false;
                        SamplingFlashLight();}
                }
            } else{ // Sampling defined from settings. Maks sampling!
                if (counter < package_size){ // save samples in buffers, less than ↩
                    package_size since else includes the last sample
                    if(counter > 1) { // check for first sample
                        xBuf = xBuf + event.values[0] + ", ";
                        yBuf = yBuf + event.values[1] + ", ";
                        zBuf = zBuf + event.values[2] + ", ";
                        tBuf = tBuf + event.timestamp  + ", ";

                    }else{ // first sample
                        xBuf = event.values[0] + ", ";
                        yBuf = event.values[1] + ", ";
                        zBuf = event.values[2] + ", ";
                        tBuf = event.timestamp  + ", ";
                    }
                    counter++; // update counter
                }else{
                    // IF-LOOP makes sure to only sample while we want to be doing ↩
                        just that.
                    if( stopsamp == 0) {
                        counter = 1;
                        csv_generator(1, xBuf + event.values[0]);
                        csv_generator(2, yBuf + event.values[1]);
                        csv_generator(3, zBuf + event.values[2]);
                        csv_generator(4, tBuf + event.timestamp);
                        xBuf = yBuf = zBuf = tBuf = null;

                    }
                }
            }
        }

        @Override
        public void onAccuracyChanged(Sensor sensor, int accuracy) {

        }

        private void addEntry(float[] accData) {
            if (counter < package_size){ // save samples in buffers, less than ↩
                package_size since else includes the last sample
                if(counter > 1) { // check for first sample
                    xBuf = xBuf + accData[0] + ", ";
                    yBuf = yBuf + accData[1] + ", ";
                    zBuf = zBuf + accData[2] + ", ";
                    tBuf = tBuf + tStamp + ", ";

                }else{ // first sample
                    xBuf = accData[0] + ", ";
                    yBuf = accData[1] + ", ";
                    zBuf = accData[2] + ", ";
                    tBuf = tStamp + ", ";
                }
                counter++; // update counter
            }else{
                counter = 1;
                csv_generator(1, xBuf + accData[0]);
                csv_generator(2, yBuf + accData[1]);
                csv_generator(3, zBuf + accData[2]);
                csv_generator(4, tBuf + tStamp);
                xBuf = yBuf = zBuf = tBuf = null;

            }
```

```
            sampleStarted = true;
        }

        public void csv_generator(int sFileName, String input) {
            try {
                switch(sFileName){
                    case 1:
                        file = xFile;
                        // start LED notification sampling. Only activated one time.
                        //if(!activatedLED){
                        //    SamplingFlashLight();}
                        break;
                    case 2:
                        file = yFile;
                        break;
                    case 3:
                        file = zFile;
                        break;
                    case 4:
                        file = tFile;
                        break;
                    case 5:
                        file = pFile;
                        break;
                    case 6:
                        file = eFile;
                        break;
                    default:
                }
                FileWriter writer = new FileWriter(file, true);
                writer.append(input + ", ");
                writer.flush();
                writer.close();

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

        //-------------------------------------------------------------------------------------------

        //                  Compute position
        //-------------------------------------------------------------------------------------------

        // TODO: 09/12/15 position has been removed to lower power consumption
        /*
        private void position(float[] accData) {

            // Extract data from acc.
            for (int i = 0; i < 3; i++) {
                acc_data[broadcastCounter][i] = accData[i];
            }

            // moving average window to low pass filter
            if (init || broadcastCounter == w - 1) {
                float x_sum = 0;
                float y_sum = 0;
                float z_sum = 0;
                init = true;
                // sum the values within the window
                for (int ii = 0; ii < w; ii++) {
                    x_sum = x_sum + acc_data[ii][0];
                    y_sum = y_sum + acc_data[ii][1];
                    z_sum = z_sum + acc_data[ii][2];
                }

                // Average within window
                x_sum = x_sum / w;
                y_sum = y_sum / w;
                z_sum = z_sum / w;

                // Classify position using classification tree
```

```
                if (Math.abs(x_sum) > Math.abs(y_sum) & Math.abs(x_sum) > Math.abs(↩
                    z_sum)) {
                    if (x_sum < 0) {
                        statusPosition = 3; // left lateral
                    } else {
                        statusPosition = 4; // right lateral
                    }
                } else if (Math.abs(y_sum) > Math.abs(x_sum) & Math.abs(y_sum) > Math.↩
                    abs(z_sum)) {
                    statusPosition = -2; // upright
                } else if (Math.abs(z_sum) > Math.abs(x_sum) & Math.abs(z_sum) > Math.↩
                    abs(y_sum)) {
                    if (z_sum > 0) {
                        statusPosition = 1; // supine
                    } else {
                        statusPosition = 2; // prone
                    }
                }
                else {
                    statusPosition = 5; // Nothing to declare
                }

                // broadcastManager
                Intent intentPosition = new Intent("Acc Sample");
                // including statusPosition to intent as Extras.
                intentPosition.putExtra("acc_Position", statusPosition);
                LocalBroadcastManager.getInstance(context).sendBroadcast(↩
                    intentPosition);
                // write position to csv file
                csv_generator(5, statusPosition + "");
            }

            // update counter
            if (broadcastCounter < w - 1) {
                broadcastCounter++;

            } else {
                broadcastCounter = 0;
            }
        }
        */
    }

    // notify that the device samples using LED.
    private void SamplingFlashLight() {
        if(samplingType == 1){
            notifSampling.ledARGB = 0x009900; // green
        }else if(samplingType == 2){
            notifSampling.ledARGB = 0xCC0000; // red
        }else if(samplingType == 3){
            notifSampling.ledARGB = 0xFFFF00; // yellow
        }

        notifSampling.flags = Notification.FLAG_SHOW_LIGHTS;
        notifSampling.ledOnMS = 200;
        notifSampling.ledOffMS = 2000;
        nmSampling.notify(17, notifSampling);
        activatedLED = true;
    }
}
```

G.2.2 XML files

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--
Created by Martin Guul 2015
Modified by Martin P. Bonnesen 14/10-2017
```

```
-->
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="master.testgraphview">
    <uses-permission android:name = "android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.RECORD_AUDIO"/>
    <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme"
        android:largeHeap="true">

        <activity
            android:name=".SampleData"
            android:label="@string/app_name"
            android:screenOrientation="portrait">
        </activity>
        <activity android:name=".Settings"
            android:label="Settings"
            android:screenOrientation="portrait">
        </activity>
        <activity android:name=".AddPatient"
            android:label="@string/app_name"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".RRSimulator"
            android:label="RR Simulator"
            android:screenOrientation="landscape">
        </activity>
        <service android:name=".AccServiceTimer"></service>
        <service android:name=".AudioRecorderService"></service>
    </application>

</manifest>
```

accsample.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--
Created by Martin Guul 2015
Modified by Martin P. Bonnesen 14/10-2017
-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <com.jjoe64.graphview.GraphView
        android:layout_width="match_parent"
        android:layout_height="200dip"
        android:id="@+id/graph1" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Sample"
        android:id="@+id/bNewSample"
        android:layout_marginTop="29dp"
        android:layout_centerVertical="true"
        android:onClick="sampleControl"/>

    <TextView
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:text="Sample nr.: "
        android:id="@+id/tvSampleNr"
        android:layout_toEndOf="@+id/bNewSample"
        android:layout_marginStart="78dp"
        android:layout_alignTop="@+id/bNewSample"
        android:layout_alignBottom="@+id/bNewSample" />
</LinearLayout>
```

activity_main.xml

```
<!--
Created by Martin Guul 2015
Modified by Martin P. Bonnesen 14/10-2017
-->

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:borderWidht="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity"
    android:focusableInTouchMode="true">

    <!-- we add graph view to display-->

    <com.jjoe64.graphview.GraphView
        android:layout_width="match_parent"
        android:layout_height="200dip"
        android:id="@+id/graph1" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Sample nr.: "
        android:id="@+id/tvSampleNr" />


    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Position:"
        android:id="@+id/tvOrientation"
        android:layout_below="@+id/bAccAudio"
        android:layout_centerHorizontal="true"
        android:layout_margin="22dp" />

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="UI Accelerometer"
        android:id="@+id/bNewSample"
        android:onClick="sampling"
        android:background="@drawable/button_green"
        android:layout_below="@+id/graph1"
        android:layout_alignParentStart="true"
        android:layout_margin="2dp"/>

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Accelerometer"
        android:id="@+id/bService"
        android:onClick="openService"
```

```
        android:background="@drawable/button_green"
        android:layout_below="@+id/bNewSample"
        android:layout_alignParentStart="true"
        android:layout_margin="2dp"/>

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Audio"
        android:onClick="audioSample"
        android:id="@+id/bAudio"
        android:background="@drawable/button_red"
        android:padding="15dp"
        android:layout_below="@+id/bService"
        android:layout_alignParentStart="true"
        android:layout_margin="2dp"/>

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Start dataopsamling"
        android:onClick="accAudioSampling"
        android:id="@+id/bAccAudio"
        android:background="@drawable/button_red"
        android:layout_below="@id/bAudio"
        android:layout_margin="2dp"/>

</RelativeLayout>
```

activity_add_patient.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--
Created by Martin Guul 2015
Modified by Martin P. Bonnesen 14/10-2017
-->
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:scrollbars="vertical"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin">
        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TableLayout android:orientation="vertical" android:layout_width="↩
                match_parent"
            android:layout_height="match_parent" >
                <TableRow>
                    <!--
                    <TextView
                        android:layout_width="244dp"
                        android:layout_height="wrap_content"
                        android:layout_marginTop="20dp"
                        android:textAppearance="?android:attr/textAppearanceMedium"
                        android:text="@string/PatientID"
                        android:id="@+id/tvPatientName"/>
                    -->
                    <EditText
                        android:layout_width="244dp"
                        android:layout_height="wrap_content"
                        android:layout_marginTop="15dp"
                        android:layout_marginRight="100dp"
                        android:textAppearance="?android:attr/textAppearanceSmall"
                        android:id="@+id/etPatientName"/>
```

```
        </TableRow>

        <TableRow>
            <TextView
                android:layout_width="244dp"
                android:layout_height="wrap_content"
                android:layout_marginTop="20dp"
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:text="@string/Gender"
                android:id="@+id/tvGender" />

            <Spinner
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="20dp"
                android:id="@+id/spinnerGender"
                android:layout_column="1" />
        </TableRow>

        <TableRow>
            <TextView
                android:layout_width="244dp"
                android:layout_height="wrap_content"
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:text="@string/Age"
                android:layout_marginTop="20dp"
                android:id="@+id/tvAge"
                android:clickable="true"
                android:longClickable="false" />

            <EditText
                android:inputType="number"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="20dp"
                android:id="@+id/etPatientAge"
                android:layout_column="1" />
        </TableRow>

        <TableRow>
            <TextView
                android:layout_width="244dp"
                android:layout_height="wrap_content"
                android:layout_marginTop="20dp"
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:text="@string/BMI"
                android:id="@+id/tvBMI"
                android:clickable="true"
                android:onClick="DialogBMI" />

            <Spinner
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="20dp"
                android:clickable="false"
                android:id="@+id/spinnerBMI"
                android:layout_column="1" />
        </TableRow>

        <TableRow>
        <TextView
            android:layout_width="244dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="@string/Snoring"
            android:id="@+id/tvSnore" />

            <Spinner
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="20dp"
                android:id="@+id/spinnerSnoring"
```

```
            android:layout_column="1" />

    </TableRow>

    <TableRow>
    <TextView
        android:layout_width="244dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="@string/Tired"
        android:id="@+id/tvTired" />

        <Spinner
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:id="@+id/spinnerTired"
            android:layout_column="1" />

    </TableRow>

    <TableRow>
    <TextView
        android:layout_width="244dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="@string/Observed"
        android:id="@+id/tvObserved" />

        <Spinner
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:id="@+id/spinnerObserved"
            android:layout_column="1" />

    </TableRow>

    <TableRow>
    <TextView
        android:layout_width="244dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="@string/Pressure"
        android:id="@+id/textView5" />

        <Spinner
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:id="@+id/spinnerPressure"
            android:layout_column="1" />
    </TableRow>


    <TableRow>
    <TextView
        android:layout_width="244dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="@string/Neck"
        android:id="@+id/tvNeck" />

        <Spinner
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:id="@+id/spinnerNeck"
            android:layout_column="1" />
```

```
                </TableRow>

            </TableLayout>
        </RelativeLayout>
</ScrollView>
```

activity_sampledata.xml

```
<!--
Created by Martin Guul 2015
Modified by Martin P. Bonnesen 14/10-2017
-->
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:borderWidht="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity"
    android:focusableInTouchMode="true">

    <!-- we add graph view to display-->


    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Position:"
        android:id="@+id/tvOrientation"
        android:layout_below="@+id/SamplText"
        android:layout_centerHorizontal="true"
        android:layout_margin="22dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="@string/sampletext1"
        android:id="@+id/SamplText"
        android:layout_centerHorizontal="true"
        android:layout_margin="22dp" />

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Accelerometer"
        android:id="@+id/bService"
        android:onClick="openService"
        android:background="@drawable/button_green"
        android:layout_below="@id/bAudio"
        android:layout_alignParentStart="true"
        android:layout_margin="2dp"/>

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Audio"
        android:onClick="audioSample"
        android:id="@+id/bAudio"
        android:background="@drawable/button_red"
        android:padding="15dp"
        android:layout_below="@+id/bAccAudio"
        android:layout_alignParentStart="true"
        android:layout_margin="2dp"/>
```

```
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/sample_data_butten1"
        android:onClick="accAudioSampling"
        android:id="@+id/bAccAudio"
        android:background="@drawable/button_red"
        android:layout_below="@id/SamplText"
        android:layout_alignParentStart="true"
        android:layout_margin="2dp"/>

</RelativeLayout>
```

dialog_bmi.xml

```
<!--
Created by Martin Guul 2015
Modified by Martin P. Bonnesen 14/10-2017
-->
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_marginTop="16dp"
        android:layout_marginLeft="4dp"
        android:layout_marginRight="4dp"
        android:layout_marginBottom="4dp"
        android:text="VÃ¦gt [kg]:"
        android:id="@+id/tvW"/>
    <EditText
        android:id="@+id/etW"
        android:inputType="number"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="4dp"
        android:layout_marginRight="4dp"
        android:hint="VÃ¦gt i kg."
        android:layout_alignBaseline="@+id/tvW"
        android:layout_toEndOf="@+id/tvW" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_marginTop="16dp"
        android:layout_marginLeft="4dp"
        android:layout_marginRight="4dp"
        android:layout_marginBottom="4dp"
        android:text="HÃ¸jde [cm]:"
        android:id="@+id/tvH"
        android:layout_below="@+id/tvW"/>
    <EditText
        android:id="@+id/etH"
        android:inputType="number"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="4dp"
        android:layout_marginLeft="4dp"
        android:layout_marginRight="4dp"
        android:layout_marginBottom="16dp"
        android:hint="HÃ¸jde i cm."
        android:layout_alignBaseline="@+id/tvH"
        android:layout_toEndOf="@+id/tvH"/>
</RelativeLayout>
```

strings.xml

```
<resources>
    <string name="app_name">MAS</string>
    <string name="action_settings">Settings</string>
    <string name="action_main">Home</string>

    <string name="prefs_Name">Name</string>
    <string name="prefs_Name_summary">Type in the patients name:</string>
    <string name="prefs_Name_default">Test</string>

    <string name="prefs_audio_fs_Title">Audio Sample Frequency</string>
    <string name="prefs_audio_summary">Type in the Fs [Hz]</string>
    <string name="prefs_audio_default">16000</string>

    <string name="prefs_audioBit_fs_Title">Audio Bit Rate</string>
    <string name="prefs_audioBit_summary">Type in the Audio Bit Rate [Bits/sample]</↩
        string>
    <string name="prefs_audioBit_default">16</string>

    <string name="prefs_audioPackage_Sample_Title">Audio Package Size</string>
    <string name="prefs_audioPackage_summary">Type in the size of audio buffer [↩
        Samples]</string>
    <string name="prefs_audioPackage_default">1024</string>

    <string name="prefs_acc_fs_Title">Accelerometer Sample Frequency</string>
    <string name="prefs_acc_summary">Type in the Fs [Hz]</string>
    <string name="prefs_acc_default">4</string>

    <string name="prefs_acc_package_Title">Accelerometer Package Size</string>
    <string name="prefs_acc_package_summary">Type in the Accelerometer Package size [↩
        samples]</string>
    <string name="prefs_acc_package_default">50</string>
    <string name="sampletext1">Start dataindsamling ved tryk af knap nedenfor. GrÃ¸n ↩
        knap betyder korrekt
        tilsluttet mikrofon.</string>
    <string name="sampletext2">Some text?</string>
    <string name="sample_data_butten1">Start dataopsamling</string>

    <string-array name="STOP_bang">
        <item>Svar</item>
        <item>Ja</item>
        <item>Nej</item>
    </string-array>

    <string-array name="gender">
        <item>Svar</item>
        <item>Mand</item>
        <item>Kvinde</item>
    </string-array>

    <string-array name="BMI">
        <item>Svar</item>
        <item>Udregnet</item>
    </string-array>

    <string name="PatientID">Dit ID nummer</string>
    <string name="Snoring">Snorker du hÃ¸jlydt?</string>
    <string name="Tired">FÃ¸ler du dig ofte trÃ¦t, udmattet eller sÃ¸vning i lÃ¸bet af↩
         dagen?</string>
    <string name="Observed">Har nogen observeret dig stoppe med at trÃ¦kke vejret i ↩
        sÃ¸vne?</string>
    <string name="Pressure">Har du hÃ¸jt blodtryk, eller bliver du behandlet for det↩
        ?</string>

    <string name="BMI">KLIK HER for at udregne BMI. </string>
    <string name="Age">Alder</string>
    <string name="Neck">Nakke omkreds over 40cm? (Skjorte krave stÃ¸rrelse)</string>
    <string name="Gender">KÃ¸n?</string>

</resources>
```

www.elektro.dtu.dk

Department of Electrical Engineering
Biomedical Engineering
Technical University of Denmark
Ørsteds Plads
Building 349
DK-2800 Kgs. Lyngby
Denmark
Tel: (+45) 45 25 38 00
Fax: (+45) 45 93 16 34
Email: info@elektro.dtu.dk