ABU DHABI UNIVERSITY

Non-Contact Respiratory Rate Measurement System Using Mobile Thermal Camera

by

Mohamed Ezat Abo yousef 1036644

Haitham Chabayta 1065457

Ahmed Sanad Ahmed 1060079

Supervised by: Dr.Luay Fraiwan

A thesis submitted in partial fulfillment for the
degree of Bachelors of Science in Electrical Engineering and Bachelors of Science in Computer Engineering

in the
Electrical and Computer Engineering department
College of Engineering

July 2020

Declaration of Authorship

We, Haitham Chabayta, Mohamed Ezat Abo yousef, and Ahmed Sanad, declare that this thesis titled, 'Non-Contact Respiratory Rate Measurement System Using Mobile Thermal Camera' and the work presented in it are our own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Haitham Chabayta

Signed: Mohamed Ezat

Signed: Ahmad Sanad

Date: 07 / 22 / 2020

"The best way to get a project done faster is to start sooner"

Jim Highsmith

ABU DHABI UNIVERSITY

Abstract

Electrical and Computer Engineering department
College of Engineering

Bachelors of Science in Electrical Engineering and Bachelors of Science in Computer Engineering

by Mohamed Ezat Abo yousef 1036644
Haitham Chabayta 1065457
Ahmed Sanad Ahmed 1060079

Obstructive Sleep Apnea (OSA) is a sleeping disorder caused by the continuous reduction of airflow while asleep. For infants', sleep apnea occurs when an infant stops breathing during periods of sleep. Sleep apnea can be detected through monitoring respiration. There are several different methods for respiration monitoring that are usually classed into contact or non-contact methods. For contact methods, the sensing device is attached to the subject's body. The non-contact methods are done through touching the subject's skin (Often done through detecting sound waves or measuring impulse). This project will investigate the use of a mobile thermal camera under a user-friendly mobile application to continuously monitor and measure the patient's respiration rate to detect sleep apnea for infants. The application firstly will detect the nose and mouth areas of the patients. The temperature distribution around the areas will be measured using a mobile thermal camera. The recorded data is sent continuously to a server for analysis to calculate the respiratory rate of the infant which will be then used to diagnose if the infant is going through sleep apnea or not. The application will then provide end-user information to the people concerned to act accordingly. It can also be used to continuously monitor the infant temperature and respiration in real-time. This will help to reduce the cost of detecting sleep apnea in comparison to the other systems often used in the real-word and will provide a simple and user-friendly way to detect a serious disorder like sleep apnea.

Acknowledgements

The completion of the project wouldn't have been possible without the support and guidance of many individuals. Firstly, we would like to express our sincere gratitude towards our professor, Dr.Luay Fraiwan for his continuous assistance and extensive guidance over the last few years. Without his support, this project wouldn't have been possible.

We are obliged to thank the entire university faculty that taught us various courses to fulfil the requirements to get awarded this degree.

We are also indebted to thank our parents who have supported us on every step of our educational journey and for always pushing us to pursue our dreams.

Contents

List of Figures

Abbreviations

SDK	**S**oftware **D**evelopment **K**it
IDE	**I**ntegrated **D**evelopment **E**nviroment
RAM	**R**andom **A**ccess **M**emory
OS	**O**perating **S**ystem
OSA	**O**bstructive **S**eep **A**penea
CSA	**C**entral **S**eep **A**penea
API	**A**pplication **P**rogram**I**nterface
RR	**R**espiration **R**rate

Chapter 1

Introduction

1.1 Problem statement

Sleep apnea is a general sleep disorder. It causes a person to pause and start breathing repeatedly in sleep. Each pause which is known as an apnea could last for a couple of seconds to a few minutes depending on the severity of the disorder. It results in the brain and the body to not receive the amount of oxygen required in order to function properly [1]. A study says that approximately 9% females and 24% males of the middle age are affected by sleep apnea in the USA. It is commonly measured by an index known as the apnea / hypopnea index or AHI which helps determine the severity of the disorder. An AHI of over 5 is considered as abnormal breathing during sleep [2]. Sleep apnea is found to be of two types; those are:

a) Obstructive sleep apnea (OSA): It is the most common type and this arises when a soft tissue collapses at the back of the throat during sleep. This results in air blockage. OSA is seen occurring in men more than women and causes them to sleep excessively during the day and also exposes them to the risk of cardiovascular diseases.

b) Central sleep apnea (CSA): In this type of apnea, the brain is unable to send signals to brain muscles to breathe although the air passage does not get blocked. This causes respiratory instability which results in paused breathing. Elderly males are more prone to CSA as some other existing medical condition may lead to the development of sleep apnea [3].

It is reported that almost one third patients with sleep apnea were involved in or nearly missed an accident due to falling asleep while on the wheel. Patients are often at a high accident risk because of the severity of their illness. Hence sleep apnea should not be left untreated especially because it can lead to many other calamities. The treatments for sleep apnea are available in different ways depending on the patient's

clinical history as well as the severity of his disorder. Many treatments require the patient to quit smoking and drinking along with discontinuing any medications which help in relaxing the nervous system. Surgical procedures can also be adopted which remove a tissue from the throat thus widening the airway. Continuous positive airway pressure or CPAP helps blow air into the air pipe, like a face mask which is attached to a machine, to keep it open. The mask is recommended by doctors when the case is very severe and conservative methods are insufficient to treat a patient [4]. Sleep apnea leads to many adverse consequences like heart disease, accidents, lack of focus, etc. it is also very closely related with obesity. Simple diagnosis is encouraged amongst people who may start facing breathing problems. The patients who have diagnosed themselves with sleep apnea must not leave it untreated as it may lead to further health complications. Sleep apnea can be detected through monitoring respiratory rate. There are several different methods for respiration monitoring that are usually classed into contact or non contact methods. For contact methods, the sensing device is attached to the subject's body. The non-contact methods are done touching the subject's skin (Often done through detecting sound waves or measuring impulse). All of these methods could be proven costly, complicated, and time wasting. So we want to come up with a solution that is applicable to anyone, user-friendly, and innovative, to simplify the process of detecting sleep apnea, decrease the cost, and optimize the user time.[5]

1.2 Motivation

Our main source of motivation to design this system came from our responsibility as engineers towards humanity and from our commitment as engineers to enhance the use of technology. We want to improve infants and their loved ones life by offering a way to easily detect such a dangerous disorder. That's why we have decided to design a user friendly system that helps patients all across the world to detect sleep apnea through just few clicks on their phones. All parents across the world will be to diagnose their infants and get alerted immediately if their infant is suffering from sleep apnea during his/her sleep to act accordingly. By designing this system we feel that we have put our knowledge to good use, by prompting an easier life to many people through the power of technology.

1.3 Background

1.3.1 Mobile Thermal Camera

A thermal imaging camera is a camera that renders infrared radiation such as light which helps to see heated areas in darkness. Thermal mobile cameras are made with materials that are heat and water resistant. They can capture body heat and are useful for a variety of health related activities. They may be handheld or mounted in helmets. Thermal cameras can help detect skin temperature variations and are also useful in diagnosing some disorders. Along with advancement in technology, thermal imaging system is proving to be a powerful solution with ability to detect temperatures. The thermal image helps in converting infrared radiation which are emitted into digital data by the target. It then visualizes the chart of temperatures through a focal plane array device. Thermal camera systems have various color palettes which can be selected by the user based on their requirement. Differences in temperatures can be best detected by the rainbow palette as it has great sensitivity of display. This color palette will prove to be useful to monitor changes in body temperature during fever as well as to keep a track on respiration for patients affected by sleep apnea. These thermal cameras come with a variety of features but the minimum feature required is that it reads the temperature of the subject. In modern diagnosis and treatment of illnesses, thermal imaging can be quite useful as it makes detection and treatment easy [6]. Physiological activities can be monitored in humans and mammals using thermal imaging systems. Different objects and living beings emit hot or cold energies of different rates. Infrared equipment is used to detect these energies which are then converted into and displayed as images [7].
Overall the achievements of mobile thermal camera are many as they provide us with new solutions for cost-effective and reliable thermal imaging. The inbuilt thermal camera on a smartphone can be useful for firefighters to rescue people or for construction workers to test the working of insulation. It can also help with medical conditions that require a patient's body temperature to be monitored regularly. In conclusion we can say that the evolution of mobile thermal cameras is making the world more reliant on smartphones.

1.3.2 Open CV

OpenCV Library is an Open source computer vision library. It is notice like a software library of computer vision applications. Furthermore, it was built in an attempt to offer a common place for applications of computer vision. Also, OpenCV used for image processing in real time. OpenCV was introduced in order to encourage companies to use the perception of machines in their products of commercial use [8].

Face recognition is not a big deal for humans, A study shows that new born babies are able to distinguish between known and unknown faces as well. We all are aware of the fact that computers are smarter, faster and better than humans hence facial recognition is accurate and reliable using a computer application system like OpenCV. It is a framework of multiple platforms and supports Windows, Linux and Mac OS [9]. OpenCV has thousands of algorithms which process and try to understand photos and videos while solving challenges. It was launched in the year 1999 by Intel. OpenCV has an application area of facial recognition which is taking over many activities of businesses. It aims to provide simulation of human eyes using advanced computer systems. It reconstructs the aspects of vision of 3D objects by extracting 2D information and analyzing it. 3D objects of real life are shown as 2D images[10]. OpenCV uses a facial detector known as Haar Cascade classifier. It captures images in the BGR format which are sized at 8-bits and the face detector classifies it as 'Face' or 'Not face'. In order to search for and detect faces from a wide variety of scales the classifier is used to processes the image several time. Such huge amount of processing takes places in a very short amount of time making the application time efficient and accurate. OpenCV's interface is primarily written in C++ though there are other interfaces like Java and Python. Facial recognition using computers has improved greatly with the introduction of OpenCV library. Also, OpenCV library is very well civilized as great results can be achieved by a code made up of an image or a couple of lines. The accuracy of face recognition can be further improved by providing more images of input taken in different angles or lighting conditions [11].

1.3.3 Android Studio

This section will present details regarding the fundamental concept related to the development of the thermal camera having Android as the base operating system. The focus will be entirely on the Android Studio which is the official environment for integrated development of android oriented applications and devices. The operating system that gets implemented in this context is the Google Android Operating System. It gets developed based on the IntelliJ, JetBrains software that is designed and developed specifically for serving the purpose of developing applications and devices on Android.
The Android Studio can be downloaded on different systems that work on different kinds of operating systems like macOS, Linux, and Windows. The IDE is regarded to be a better form of replacement for the Eclipse-based tools for Android development. This is where the native Android oriented developments used to take place. The announcement of the introduction of the Android Studio took place in 2013. As time passed by several updates have been released amounting to around 8 different kinds. From 2013

to 2019, the IDE had Java as the backend programming language. After that Java was replaced with Kotlin which is a further advanced language to provide further flexibility and opportunities to the users in developing innovative and new programs. However, C++ and Java are still used as the supporting languages and add-ons to the IDE. In going into further detail corresponding to the Android Studio, quite a few features have been highlighted that can be directly linked to the application. Those features are as follows:

- The build support of the IDE is based on Gradle.
- It facilitates the developers with hotfixes and refactoring parameters that are specific to the Android platform.
- Different kinds of Lint tools remain present for catching the performance of the processes along with version compatibility, usability, and other related problems to develop.
- It acquires the potential of app-signing and ProGuard integration.
- Several templates exist in it which is regarded to be the wizards that aid in designing the common components related to Android.
- It provides a fantastic and rich editor for editing the layouts of the applications and devices. It facilitates the users to design the User Interfaces using just the facility of drag and drop.
- The previews of the layouts can even be seen on configurations of multiple screens as well.
- It provides support and assistance for building applications related to Android Wear.
- It has the support of built-in nature for the integration of Google based Cloud Platform. It enables proper integration with the app engine of Google and Firebase Cloud Messaging.
- It acquires an emulator or virtual device that facilitates running and debugging applications on the Android Studio itself [12] .

The Android Studio gives developers a massive level of openness and opportunity in the context of development. The major reason behind it is that the IDE allows the developers to write the programs almost in all kinds of programming languages. It acquires compilers and supports different kinds of programming languages falling under

CLion and IntelliJ. It supports almost all the versions of Java and C++ language through which the programmers can write the codes as per necessity. Moreover, the use of the Kotlin language for writing programs is also supported by the IDE [13]. The following section will present details regarding the system requirements that are needed by the Android Studio to run smoothly in the computer-based devices. It can be seen that the amount of system requirement can be regarded as negligible in comparison to the level of output the IDE generates to the world. The three most used operating systems and the relevant hardware as well as software requirement will be highlighted in the following section. In the case of Windows, the requirements are Windows 10 or Windows 8, or minimum Windows 7. The minimum is 64-bit support in the case of running an Android Emulator. Considering the Mac OS, the 10.10 version is needed or maybe higher in comparison as per macOS Mojave. Focusing on Linux KDE or GNOME also gets included in this aspect. The common requirements for all the operating systems are as follows:

- In the case of RAM, a minimum of 4 GB of RAM is required but recommended is 8 GB RAM.
- The necessary disk space required for this matter is 2 GB of minimum space in the hard disk. The recommended is 4 GB.
- The reason behind it is 500 MB linked to IDE and 1.5 GB for the Android SDK. Moreover, a large emulator system remains present in a clear approach.
- The platform or framework of Java Development Kit 8 gets implemented.
- The minimum resolution in terms of the screen is 1280 X 800 [14].

1.3.4 Measurement of respiratory rate using a thermal camera

This section will present details regarding the respiration rate that gets measured by the thermal cameras developed on the Android platforms and get integrated within the mobile cameras. The respiratory rate refers to the rate at which human beings breathe. The rate mainly gets measured based on the number of breaths done per minute. This gets set as well as controlled by the respiratory center. This gets measured by the number of breaths a human being completes within a minute. In general, the measurement of the respiratory rates in the case of humans gets measured by counting the total number corresponding to breaths in the context of a minute oriented basis. It gets done by counting the different times, the chest of human beings get raised. At present, the doctors make use of the fiber-optic sensor in terms of breath rate. This can

be implemented to monitor the patients in terms of scanning the images of magnetic resonance. The respiratory rates may get increased during the times of illness, fever, and other harmful medical conditions. However, the major drawback that has been identified in this context is inaccuracies in the measurement of the respiratory rate [15]. To mitigate the issues linked to the measurement of respiratory rate, the application of advanced forms of technology like Android has been incorporated in a clear approach. The application of the thermal camera can lead to the generation of accurate measures in a clear approach. Respiratory rate (RR) is a significant indicator corresponding to health presently the measurement of RR majorly depends on the methods of contact with inherent levels of difficulties like discomfort for patients, complications in setups, and the tendency corresponding to errors linked to dislodgement of movements [16].

It can be noted that the exhaled air remains always warmer in comparison to the air that gets inhaled by human beings. Both of these parameters impact the temperature of the skin that remains surrounding the nose. How these fluctuations take place remains quite evident when the thermal camera gets incorporated. The locations also get analyzed for understanding the pattern of respiration among the individuals. It can be noted that the rate of respiration in the case of infants has been found to remain more than 80 different cycles per minute.

The thermal camera working on the platform of Android has been developed for processing the thermal and infrared images at a static rate having 25 frames occurring every second. But the most effective parameter that can be highlighted in this aspect is that the major factors of inaccuracy in the case of measuring the rate of respiration are that the thermal camera completely mitigates the impacts of movements that encompass the heartbeat that takes place in a clear approach [17].

In the case of the application of thermal cameras for Respiratory rate measurement, the application of a LabVIEW can be incorporated in the application development. This provides us with the ability to control the acquisition of data, process images, and to display the RR metrics and images. There are different stages for determining the rates of respiration among human beings are as follows:

- **Acquisition of image and segmentation of subject**
  The camera captures images with the help of NI Vision. We also implement the Flir one based LabVIEW toolkit integrated with ThermoVision into our system. A high-pass filter is used to remove the impulsive noise. The images are also reduced to 8 bits and then it gets segmented to remove the background objects to get the subject alone. [18].

- **Detecting and tracking the face**
  The image for subjecting gets further segmented for capturing only the face without any kind of extra details regarding the body functioning of human beings. The face only gets mapped as per the image that gets acquired originally for critically indicating the detection [18].

- **Detection of the eye corner**
  Through the thermal camera operating on the Android platform, a specific corner of the eye gets concentrated and eyes are found to be the warmest points in terms of the faces. It remains located and the position and value enable in figuring out the location at the corner of the second eye. The major role played during this situation is the intelligent algorithm of learning. It gets developed by the development module of NI Vision [18].

- **Detection of the nose and extracting the rate of respiration**
  This is the final stage where the corners of the eyes facilitate the camera in figuring out and locate the nose tip of human beings. It is the coldest portion lying within the face. The coordinates that get figured out using this, facilitate in defining a proper region of interest (ROI). The ROI remains to be at the nostrils right below both corners corresponding to the eyes. The values of pixels that lie within the ROI get processed and extracted as per respiration featuring the overall frame. The operation gets repeated and each of the frames incoming, and lastly, the processing signal performs the task of calculating the respiration rate [18].

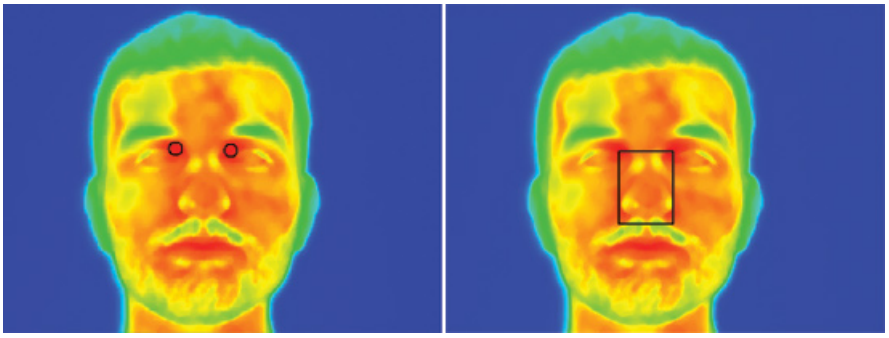

FIGURE 1.1: Detection of the the nose by thermal camera [18]

1.3.5 Importance of respiratory rate measurements

The respiration rate(RR) is a vital parameter and most respiration measurement methods are proven to be costly and complicated. The search for a better RR measurement technique is is essential with the continuous demand for better techniques in the medical

field. RR is the total number of breaths taken by human beings within a single minute. It is a clinical signal that represents the factors of ventilation and it corresponds to air that goes in and out the human lungs. Fluctuations in the RR is if often regarded as an initial sign of a breathing disordered as it could occur because if the extra effort the body puts into maintaining the delivery of oxygen into the tissues and it could damage them and cause deterioration. [19].

On failing to recognize the initial signs corresponding to deterioration, it can result in further damages to the health conditions of patients. Upon reviewing the abnormal fluctuations via measuring the RR, it can be used to find out people who are suffering from adverse and serious kinds of diseases could have been diagnosed and treated early by measuring the RR 24 hours before the day when the patients get diagnosed by prominent symptoms. The failure in appropriate respiration turns out to be the primary parameter in terms of admission to the intensive care unit. Further more, the enhancement of RR can be aided in predicting the patients who remain at risk of having a cardiac arrest. A common example of that is a study of retrospective nature demonstrated that if the RR measured is greater than 27 breaths per minute is a prominent predictor that a human being is prone to a cardiac arrest within a time duration of 72 hours [20]. Further important aspects that can be highlighted in the case of measuring the RR are the fundamental portions that it provides to asses the health condition of patients and that can be used to determine the improvement in patients health conditions.[21].

The normal RR for adults at rest is from 12 and right up to 20 breaths per minute. If the RR remains less than 12 or greater than 25, then it is considered to be an abnormal RR. If the RR falls into the abnormal range it may represent asthma, pneumonia, anxiety, or congestive condition for patients. It has also been often reported that abnormal RR could represent different lung diseases. In addition it can also represent narcotics, or even overdose of drugs [22]. RR could help with diagnosing numerous diseases and disorders and here is the common diseases that can be identified in through measuring the RR:

- Dyspnea
- Apnea
- Tachypnea
- Hyperpnea
- Bradypnea
- Hypopnea
- Bradypnea

- Platypnea
- Orthopnea
- Respiration linked to Cheyne-Stokes
- Biot's respiration
- Kussmaul breathing

1.4 Literature Review

This section will present a detailed description of the approaches and methods that are implemented currently in real-world applications for measuring the respiration rate and the problems present in each of these methods and how our proposed solution could help with preventing these issues.

1.4.1 Optimal quantization

In an overview, it can be drawn that optimal quantization is about mapping the elements of temperature for the digitalization of the pixels based on color maps for different thermal ranges of dynamic scenes. It is the method or procedure that facilitates translating the values of continuous temperature for its digitally equivalent colored maps. It can be demonstrated through elaborating on the following algorithm. The range lays within the limits of T0 to Tk-1 and the color pixels that turn out to be the most essential aspect of this method ranges between u0 to UK-1. The standard equation that gets implemented in this aspect is u = (T). This is applied to map the coordinates between the ranges appropriately [23]. In the context of quantization in the thermal dynamic range of scenes as per variations in time, the adaptation gets done as per quantizing the distribution as per thermal nature by sequencing the thermal range and finding of the interests in a clear approach containing the entire distribution of thermal sequences via finding out a thermal range as per interest containing the entire distribution of facial temperature in all the single frames. Initially considering the first step, the statistical extremity in terms of removal of value and processing and reducing the unexpected noises take place. The common example is the protection of sunlight in the glasses of individuals. Furthermore, the points corresponding to extreme temperature can be produced using reckoning the errors in the context of thermal imaging via mobile devices. In this aspect, the calculation of temperature gets done via the emissivity of fixed nature or utilizing the conditions of the lens. A common example in this aspect can be highlighted to the misted lens. The process gets executed by the removal of all kinds of thermal signals

and it is set beyond the standard deviation of 1.96. The initial candidate that gets set to be the thermal range is from T'min to T'max. The equation that can be presented in this aspect has been presented in the following section:

$$T'_{\min} = \bar{c} - 1.96\frac{\sigma_c}{\sqrt{n \cdot m}}, \quad T'_{\max} = \bar{c} + 1.96\frac{\sigma_c}{\sqrt{n \cdot m}}$$

FIGURE 1.2: Initial candidate for the thermal range [24]

In this equation, 'c' is the sample mean encompassing c(x) is the distribution of temperature of one-dimensional nature. The distribution in this context gets represented as (x1, n m). As per the presented aspects, n.m is the spatial resolution consisting of collected matrixes of thermal distribution. In addition to it, c gets regarded to be the standard deviation of the overall parameter of c(x). Moving on to the final section of this method, several ranges corresponding to interest are assumed to show that there are two different elements of qualitative nature in each of the different frames. The initial frame is the background, and the second is the face of the person. This method figures out the threshold value too that aids in separating the background from the main object. This gets done via the analysis of the color oriented histogram. This can facilitate searching the thermal values by distinguishing the human skin from different areas of skin from the time-varying histograms in the case of temperature. This exhibited several ranges of dynamic characteristics. Moreover, it has also been presented that this method is quite robust even in the case of the non-bimodal presence corresponding to histogram as well. This leads to finding an optimal boundary quite difficult in the aspect of different ranges of temperature interests. The technique of Optimal Quantization gets named after its concept and gets finalized based on the iterative computation of the complete threshold value Top as presented in the following section. [24].

$$T_{opt}(0) = T'_{\min}$$

$$T_{opt}(t+1) = \frac{\mu_1(t) + \mu_2(t)}{2}$$

FIGURE 1.3: Iterative computation of an optimal threshold value T [24]

Based on the above equations, u1 (t) and u2 (t) are the mean values if c (x) $\leq Topt(t)$ or c (x) > Topt (t) respectively. The method that has been represented requires the coordinates of four corners of the image for containing the background pixels. The process gets further iterated until the equation Topt (p) - Topt (p-1) 0 is achieved. This is where the ranges of temperature related to interest get chosen as:

In the cases of accomplishing the average temperature in the face of background, it includes the air and hair of individuals. It remains lower than the cutaneous skin of human beings. This method only concentrates on the lower bound in terms of determining the interest of the optimal range. During the situations when the average temperature remains more than the background then the use of the upper bound and the skin of the human cutaneous portion also get done [24].

1.4.2 The flow of Thermal Gradient: tracking the nostril-region

This method mainly emphasizes on the region around the nose. To accomplish the robust performance in terms of tracking the nostril, a new algorithm has been proposed for making use of the thermal camera to measure the rate of respiration. This algorithm has been presented as the Thermal Gradient flow. Different kinds of colored images get generated using optimal quantization and this algorithm performs the task of computing the magnitude of thermal gradient via the implementation of the matrix. In every frame, the application of the Median flow-oriented algorithm gets done that makes use of the error estimation done by forwarding and backward tracking. In terms of leveraging the robustness, the compensation of the loss of points related to feature is done via resting the ROI oriented to the two-dimensional gradient-based cross-correlation of normalized nature. The diagrams below present the different colored images that are obtained by this application [24].

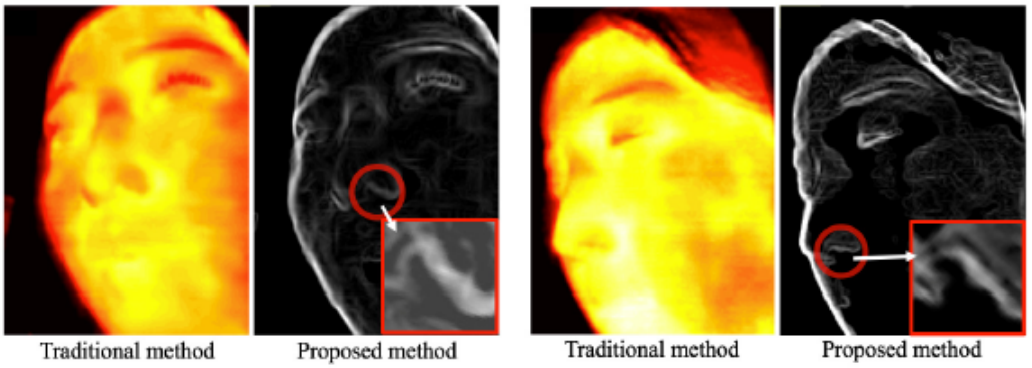

FIGURE 1.5: Shots of conversion from thermal images to thermal-gradient magnitude maps
[24]

The metabolism aspect as per the human homoeothermic factor and the relatively low level of thermal conductivity linked to the human skin perform the act that gets done by the filters of low pass characteristics. This is the reason why the shapes of the nostril and nasal areas remain blurred. This action results in generating differentiation of the weak nature of the key features at the facial point. In the context of figuring out clearer features, the boundaries get enhanced between the ala region of the noses and the nostril

by converting the thermal image having quantized features. The thermal image (u) gets implemented within the magnitude map oriented thermal gradient of two-dimensional characteristics in the following approach.

$$\Phi(x,y) = \sqrt{\left(\frac{\partial u(x,y)}{\partial x}\right)^2 + \left(\frac{\partial u(x,y)}{\partial y}\right)^2}$$

FIGURE 1.6: The two-dimensional thermal-gradient magnitude map [24]

Based on this equation, x as well as y coordinates within the x-y plane within the space of the image. A further distinct nature morphological shape of the nostril can be accomplished in the form of an image via the use of a thermal gradient in comparison to the normal thermal image. The combination of several artifacts like respiration dynamics and motion gets incorporated as well. The image that gets converted can then be made use for collecting different points having a feature that represents the tracked motions of the nostril [24].

The map of the thermal gradient within the nostril-ROI, ROI, is chosen via the selection of nostrils in the form of ROI based on the size ranging from N X N pixels within the first frame. This can be presented in a manual approach either automatically or via human effort. Based on the point tracking, the making use of the algorithm linked to Median Flow also gets done. It is because it has facilitated in proving the performance of tracking in the context of thermal imaging in a sector of the non-biomedical sector. The following algorithm will aid in calculating the error in terms of backward and forward errors defined in the following manner [24].

$$e(T_f^k \mid S) = \left\| x_t - \hat{x}_t \right\|$$

FIGURE 1.7: Forward-backward error defined algorithm [24]

In this case, the image of thermal gradient sequences is the following approach, S = (t, r+1, ..., r+k), the trajectory in the case of forwarding track is oriented to Tfk (xt, xt+1,..., xt+k) and the trajectory in the context of backward trajectory Tbk = (x't+k, x't+k-1,.....,x't) that get produced based on the backward tracking in the basis of the first frame. In this case, xt+k along with —xt-x't— gets denoted to be Euclidean distance oriented to two different points. Considering the algorithm, the points get tracked through the implementation of the KLT. In this case, the points get selected based on nostrils ROI based on the magnitude of the thermal gradient as per the map. Based on the final strategy that can be handled for the case of the point-based features

that get completely lost that gets implemented as per the two-dimensional correlation-based normalized cross based relation [24].

In specific aspects, the combination corresponds to correlation in terms of gradient as per the shown perspective for generating high levels of performance in the case of registering the deformable components in terms of neuroimaging. In a similar aspect, the tracking of performance gets further enhanced in terms of nostrils via searching for a new position within the ROI. This maximizes the overall normalized coefficient based cross-correlation as per gradient.

$$\gamma(\mathrm{x}) = \frac{\sum_i (\Phi_{ROI}(\mathrm{x}_i) - \mu_{\Phi_{ROI}(\mathrm{x}_i)})(\Phi(\mathrm{x}_i - \mathrm{x})\mu_{\Phi(\mathrm{x}_i - \mathrm{x})})}{\sqrt{\sum_i (\Phi_{ROI}(\mathrm{x}_i) - \mu_{\Phi_{ROI}(\mathrm{x}_i)})^2 \sum_i (\Phi(\mathrm{x}_i - \mathrm{x}) - \mu_{\Phi(\mathrm{x}_i - \mathrm{x})})^2}}$$

FIGURE 1.8: New position of the ROI
[24]

As per the above equation, x is the set comprising (x, y) right at the center of the square represented by N x N. This size remains similar to the area of the ROI. In the situation of numbering the tracked points and this falls under the overall value of the threshold. This method aids in resetting the overall ROI and finding out new features as per the gradient. Also, This method gets applied via the application of the selection of ROI automated aspects. This is done right at the initial frame in the content of gathering several sets of images of the nostril [24].

1.4.3 Estimation of the estimation rate via the integration of Thermal Voxel

The exchange of heat that takes place within the nostrils during the times of exhalation and inhalation gets determined through the pattern of breathing for an individual. In terms of monitoring these patterns, the methods that exist generally and end up computing the average value corresponding to the distribution linked to thermal distribution within the nostril ROI in all the frames that turn up. This mainly seeks for the fluctuating patterns on a timely basis.

On the other hand, it can be explained that several issues make use of the average distribution that takes place. Hence, in this aspect, it can be highlighted that there are quite a few issues that make use of the average distribution that takes place in this context. This is where the introduction of the Thermal Voxel of three-dimensional nature further aided in enhancing the quality corresponding to the signals of breathing. This gets inspired by the voxels within the neuroimaging. This approach mainly maps all

the two-dimensional thermal units within the three-dimensional space of the voxel. This feature turns out immune to different kinds of factors for inducing the low quality of breathing as per information and pointing out the global changes within the distribution of spatial thermal as it focuses on the task of extracting the breathing that gets induced as per the changes in the thermal volume within the nostrils by computing the quantity within the thermal voxels [24].
The construction gets done in the following manner using the formula presented below. In this formula, where t (T) represents the integral aspect of the thermal voxels within

$$\hat{v}(t) = \int_{T_{min}(t)}^{T_\delta(t)} \Lambda_t(T)dT \approx \sum_i \sum_j T_\delta(t) - \hat{u}_{ij}(t), \quad T_\delta(t) - \hat{u}_{ij}(t) > 0$$

FIGURE 1.9: Integral of the thermal voxels
[24]

the nostril as per cross-section remaining having temperature T. u'v is the temperature of absolute nature that gets tracked within the region. Tg signifies the upper boundary for integrating the concave volume that gets set for a moving average of temporal characteristics. In this case, the moving average of temporal nature gets considered to be n = 2. It gets calculated as per the spatial temperature oriented mean. The objective remains to acquire a stable boundary along with the changes in the global thermal aspect. In the case that the ROI tracker loses the control of tracking of the region of nostrils, Tg would be required to be reset right from the following frame for rejecting the value from the bounding box of misplaced characteristics. The displacement can be detected based on inspecting sudden and several changes which differential and statistical skewness are corresponding to the thermal distribution.

$$\Delta\gamma_1' = E\left[\frac{\hat{u}_{ij}(t) - \mu_t}{\sigma_t}\right]^3 - E\left[\frac{\hat{u}_{ij}(t-1) - \mu_{t-1}}{\sigma_{t-1}}\right]^3.$$

FIGURE 1.10: statistical skewness on the thermal distribution
[24]

The following section will present details regarding the different extraction patterns of respiration by the integration of Thermal Voxel. The initial representation that will be presented in the following section corresponds to the nostril of a human being and its different sequences within the thermo gram.

The next image that gets captured by the thermal camera corresponds to the concave oriented volume that varies based on the variances in heat within the nostrils and the extracted signals of the respiratory by comparing it to the signals of ground truth.
Furthermore, the final process that gets executed in this method is the comparison

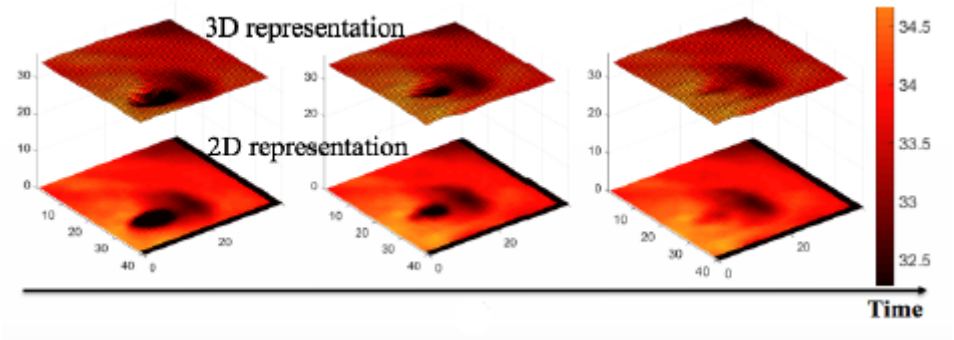

FIGURE 1.11: Different patterns of respiration by the integration of Thermal Voxel [24]

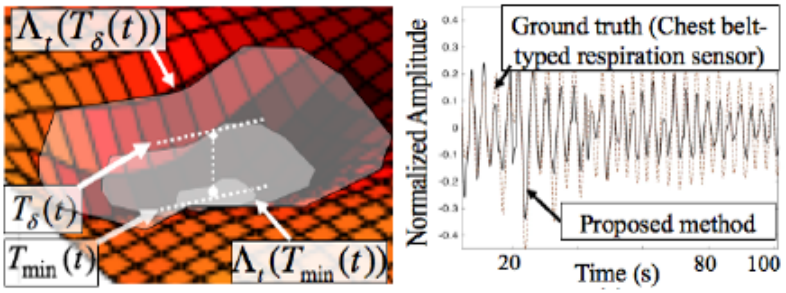

FIGURE 1.12: Variances in heat within the nostrils and the extracted signals of the respiratory [24]

linked to voxel oriented signals in a filtered manner with the method that is traditional in nature. It entirely depends on the changes done by the participants in their heads. The voxel approach critically tracks the ground truth that the traditional approaches fail to do. In the context of measuring the rate of respiration, the application of the time domain and the frequency domain gets done. The frequency-domain corresponds to the detection of peak frequency.

On the other hand, the time domain emphasizes the Bayesian approach oriented to the estimators of a short-time manner. Generally, the application of the power spectral density of short term nature gets implemented. This entirely analyses the self-similarity feature of the thermal voxel, v' (t) for determining the rate. This gets accomplished through computing the Fourier transforms Ff corresponding to the short-term function of autocorrelation. In terms of decreasing the ripples within the domain of frequency due to the existence of truncated windows of short-term nature, the implementation of the Gaussian window gets done in the following manner [24].

The length of the window gets determined as wi (k) remains to be 2t'fs + 1. In this aspect, t'max remains to be the upper limit corresponding to respiration. fs is the frequency of sampling. The value linked to t'max gets determined to be the minimum

$$w_i(k) = \hat{v}(n_i + k)e^{-\frac{1}{2}\left(\frac{k}{\sigma}\right)^2}, \quad k \in \{-\hat{t}_{\max} f_s, \ldots, \hat{t}_{\max} f_s\}.$$

FIGURE 1.13: Gaussian window
[24]

expected rate of respiration related to interest. In this approach, the use of similar ranges of the expected rate of breathing remains within the range between 0.1 Hz to 0.85Hz. Hence, t'max is equal to 10s. Once wi (k) gets computed, the value remains in a normalized nature in terms of featuring on the scale. In addition to it, the output remains filtered via the filter of third-order elliptic nature. It includes a stop band attenuation linked to 6 dB and a passband ripple lying around 3 dB. The frequency of the pass band cutoff remains around 0.1 Hz along with 0.85 Hz. Considering the above-mentioned parameters, the estimation of the respiratory rate via searching frequency remains to be around frequency f that maximizes the overall spectral density of power.

$$S_T(f) = F_f(R_{ww}) = \sum_k R_{ww}(k)e^{-j2\pi fk}$$

FIGURE 1.14
[24]

In the equation above, Rww represents the autocorrelation of short-time within the filtered wi (k) [24].

1.4.4 Remote monitoring of the dynamics of breathing through infrared thermography

This is another method that gets implemented to calculate the rate of respiration by making use of the thermal camera. This camera runs on the Android platform and acquires the mechanism of working with infrared rays. It has an irregular frequency of respiration that gets considered to be of the earliest markers corresponding to physiological distress. Furthermore, monitoring the vital aspect plays a significant role in the context of diagnosing different kinds of diseases that take place in human beings. Any kind of disorder in the pattern of respiration gets caught through this mechanism. The method that gets implemented comprises different steps:

- Detecting the ROI automatically
  The initial step that is considered is automatically detecting the ROI. It is done by the implementation of Otsu's method of multilevel nature. The incorporated algorithm implements the discriminant analysis for estimating a threshold value

of optimal nature. In the second step, removal of the background noise gets done and the region having the largest area coincides with the binary image by linking to the face of the subject and the final step within this aims to locate the contour of the chin within the face [25].

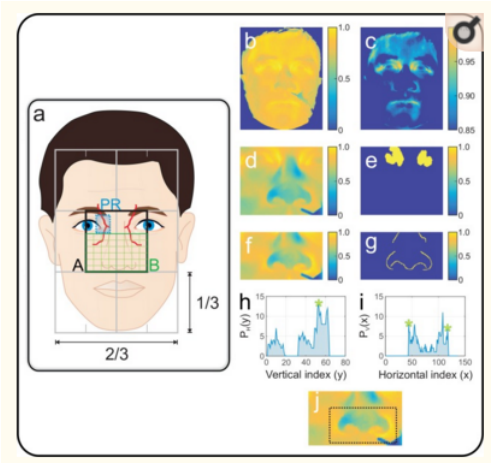

FIGURE 1.15: Steps to find the ROI [25]

- Tracking of the ROI

  The algorithm of tracking is the research that gets done using sparse representation. The particle framework via filter gets integrated into a clear approach. The filter particle consists of two different models. The initial model is the model of state transition and an observation model and getting denominated through appearance model. In this case, two different aspects get considered. The initial thing is conditional density getting represented as p (xt—xt1).The next aspect of getting considered is the observation model. It performs the task of estimating the similarity between the target model and the target candidate [25].

- Identification of the concept of ROM

  In terms of improving the ratio of sound to noise, a smaller and second ROI, and denominated the ROM gets defined. It gets computed for each of the ROI that gets tracked. The ROM gets defined in considering the edges of the human's nose. They get figured out utilizing the implementation of the canny edge detector. Also, vertical and horizontal projections get calculated in a clear approach [25].

- Extraction of the signal processing and breathing waveform

  In this step every frame acquires the value of mean temperature s' (t) in the case of calculating ROM gets calculated in the following manner via the implementation of the equation:

  In this case, s (i, j, t) in the measurement of image temperature corresponding to

$$\bar{s}(t) = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} s(i,j,t),$$

FIGURE 1.16: The mean temperature value s(t) of the ROM [25]

the pixel (i, j) and the time point m, t and get described as per the width of the ROI. The final aspect is n representing the length. The breathing signal relates to the mean temperature within the frames of the ROM. The signal had been further processed via the application of a second-order filter as per bandpass with an upper and lower cutoff of 3 dB frequency range between 0.1 Hz and the 0.85 Hz in a respective manner. It also facilitates the aspect of instantaneous breathing. In this, every step gets calculated in a clear manner and approach. This is where several approaches get considered clearly as per this approach. Autocorrelation of the adaptive window gets calculated and followed by the considered window average of an adaptive nature and differences in terms of difference function gets considered, EAMDF[m]. The next is pairs of maximum amplitude EMAP[m], and the third is the autocorrelation of adaptive window EAC[m]. These three aspects get combined based on three different estimators, P (m — EAC, EAAMDF, EMAP) [25]. In this case, the equation that gets implemented is as presented below:

$$P(m|E_{AC}, E_{AAMDF}, E_{MAP}) \propto P(m|E_{AC}) \cdot P(m|E_{AAMDF}) \cdot P(m|E_{MAP}).$$

FIGURE 1.17: Bayesian fusion method [25]

1.5 Impact Statement

This section will present details regarding impact corresponding to the measurement of the rate of respiration by making use of the mobile thermal camera within the environment, society and economy. It has been figured out that thermal mobile cameras get implemented for measurement of respiration and these impacts three different dimensions. The initial is the environment, and the next is society, and the last factor is the economy.

1.5.1 Environment

It has been reckoned that the use of mobile thermal cameras for measuring the rate of respiration impacts positively on the environment. It can be noted that the thermo graphic cameras within the mobile phones and working on the Android platform performs its operation by detecting infrared radiation. Through this process, the images get displayed where the colors directly correspond to the temperature of the nasal region of human beings. It also encompasses details regarding the chin, throat, and the corners of the eye as well. The technology has several applications like identification of the fluctuations in temperature within the face that remains linked to the respiration rate and making use of the collected data for generating accurate outcomes [26].
These cameras are highly essential for the environment because it acquires the power of tracking gases that are dangerous for human beings and the environment. These gases remain invisible to the naked eyes. It has been reckoned that different kinds of emissions take place regularly all over the world. This may affect the local environment and the climate. As a result, human beings breathe this harmful air in and become prone to different kinds of chronic diseases. It gets regarded to be a form of slow poisoning of human beings. In addition to it, several gases are quite inflammable in nature and acquire the capability of resulting in a fire hazard. In these aspects, the mobile thermal cameras turn out to be highly essential for tracking those gases using the inbuilt infrared tracker in it [27].

FIGURE 1.18: FLIR ONE
[27]

The mobile-based thermal camera facilitates in detecting these gases by focusing on the environment of a particular area and capturing a picture. This camera figures out the presence of harmful gases like greenhouse gases, SF6, methane, and carbon dioxide.

These gases get caught within the images captured by the camera, and adequate measures for the leakage of those get taken. This helps in stopping the gases to get mixed with the atmosphere and end up polluting it. As a result this also facilitates the people living in it as well. These cameras prevent humans from breathing in harmful gas. Another aspect that can be highlighted in this context is that these mobile-based thermal cameras also positively indirectly influence the environment. Using the ROI calculation, these cameras figure out any kind of abnormalities in terms of the rates of respiration among human beings. In this aspect, it can be noted that the cameras even end up tracing whether the cause of the abnormality in terms of respiration rate gets caused due to the presence of any kind of harmful gas in the body or the atmosphere that particular person resides. If any traces of harmful gases get figured out adequate measures are taken and this cures not only the human body but also the environment in a direct approach and manner. As the mobile-oriented thermal camera can figure out differences in temperature through different kinds of heat maps, it is also able to catch any kind of harmful infectious germs that are existing in the atmosphere as well. It acquires the ability to catch images of the presence of different kinds of microorganisms and by the temperature differences; the harmful ones can also be identified. This ability impacts on the environment in a further positive way, since it generates awareness regarding the existence of germs, or in severe cases, it gives signal regarding the rise of a pandemic. This is where both the humans and the environment get highly benefited through the Android oriented thermal camera. As a whole, it can be presented that the procedure of measuring the rate of respiration by the mobile-based thermal cameras positively aid the environment since the camera acquires potential to figure out several factors that lead to degradation of the environment. It generates awareness so that people can consider measures for betterment [27].

1.5.2 Society

In the context of thermal imaging oriented mobile cameras working on the Android platform for measuring the rate of respiration among human beings, it can be demonstrated that it has led to a major transformation corresponding to society. The major reason behind it is that society can be directly linked to the people living in it. Considering this point, it can be highlighted that the thermal camera highly benefited people working within the industry of medical science specifically to conduct the operations of diagnosis and medical tests at ease.

It has been presented that previously during the time of measuring the rate and pattern of respiration of humans through traditional devices, different kinds of inaccuracies could be noted. This is the reason why doctors generally did not focus on the respiratory rate

measurements for prescribing medicines or treatments. However, they always believed that the respiration rate serves to be a major signal of any kind of damage within the bodies of human beings. Since there had been no appropriate mechanisms or devices for accurate measurement of the rate so they had to compromise this aspect and considered the alternative solutions. This is where the thermal cameras even in the mobiles have turned out to be a boon for the doctors and as a whole for the entire society. At present, these mobile thermal cameras get implemented throughout the industry of medical science and the doctors find it highly fruitful to acquire an accurate measure of the respiration rate and they can proceed with the treatments without any delays. The device is found to be highly safe and simple to use. No emission takes place from the camera and nothing gets injected within the bodies of the patients for having the readings. It does not do any kind of damage to the fragile DNA as well. It has facilitated faster and quicker treatments of diseases in humans [16].

Further advantageous parameters of the mobile camera working on thermal parameters to the society are that it does not generate any kind of radiation into the bodies of human beings and it is completely free from any kind of pain. The camera does not make any kind of contact with the bodies of human beings for measuring the rate of respiration. No need for compression is also there in this context. The thermal camera is quite cost-effective in comparison to the other procedures for diagnosing different kinds of images. It can also be made use in the cases of men, children, and women and generates the accurate rate and pattern of respiration and beating of the heart. It turns out ideal in the cases of preventive medicine. This is because it facilitates detecting the diseases in the human body right at an early stage. After all, the first symptom of any kind of disease can be observed in abnormalities in breathing patterns and the rate of respiration. This helps the doctors in making timely investigations and they intervene before the symptoms getting manifested. The thermal camera further positively influences society in terms of screening of health in an appropriate manner. The aspect of thermography generates unique levels of insights regarding the physiological aspects of human bodies. It even facilitates in acquiring appropriate knowledge and ideas regarding the health conditions of human beings via the measurement of the respiration rate and the pattern of breathing right at the first glance in the aspect of developing conditions. Otherwise, the diagnosis process may take up a huge amount of time and as a result, leads to major negative outcomes. The mobile-based thermal camera serves highly beneficial purposes for society by highlighting the aspects of healthcare and medicine in a clear manner and approach. It gets implemented in the practical field in a clear manner and approach. The thermal camera gets implemented within the practical field like spotting different kinds of fevers along with anomalies linked to temperature. This has proven out to be quite important in medical screening as well.

The mobile thermal cameras can accurately and quickly scan all the patterns of breathing and the rate of respiration. The recent outbreaks corresponding to Ebola and SARS can be diagnosed directly and quickly by the breathing differences and turn out to be highly crucial in screening and taking the necessary measures. In addition to it, the thermal imagers have provided quite a few diagnosed ranges related to disorders that remain associated with the activities of different kinds of parts of the body [17].

1.5.3 Economy

It has been obtained in the previous sections that the procedure of measuring the respiration rate via mobile thermal cameras has been quite fruitful for the environment and society. It has been reckoned that the use of thermal cameras is going on increasing within the medical industry all over the world. This is because it facilitates in executing the tasks of screening the diseases among human beings through the patterns and rates of breathing as seen among the patients. Furthermore, it also aids in tracking the dangerous gases in the atmosphere due to its infrared mechanism and the advanced Android platform integrated with artificial intelligence. Environment, as well as society, is two significant aspects and governments all over the world are considering different kinds of measures for the well-being of both these two aspects. Hence, the demands for mobile-oriented thermal cameras are going on increasing. It has turned out to be a significant aspect and is bought by every medical organization at present. This is where the organizations manufacturing these cameras acquire a significant margin of revenue and as a result impact the economy of the country distinctly. Trade operations in the form of export and import are also taking place from one country to the other and this has been generating major growth in the world economy as well [28].
After the outbreak of the novel COVID-19, thermal camera have turned out to be more popular since the screening of individuals can be done quite easily and quickly through this. The modulations in the heat maps and temperature fluctuations near the nostrils provide the signals whether a person is a victim of the corona virus. Therefore, several places like airports, bus stops, shops, shopping malls, supermarkets, etc. are buying these cameras to do an initial screening of the passengers and the customers to figure out their health conditions. The cost of thermal-based mobile cameras is not much, but it is getting sold in thousands of numbers and resulting in the generation of a huge economy. Also, it has been figured out that the doctors are focusing on precisely measuring the rate of respiration for diagnosing the presence of any kind of harmful substances in the bodies. Any kind of abnormalities within the pattern of breathing can be regarded as a signal of some kind of problems. Moreover, these cameras also consume very little time and do not require any kind of contact to generate the results. The thermal

cameras used in the medical industry having significant features are a bit expensive in nature. But the potential it acquires and the lack of alternatives have facilitated the manufacturing countries to earn a significant profit and as a result, influences positively on the economy [29].

Chapter 2

Design

2.1 Problem description

Respiratory rate measurement is vital in the medical field and the methods currently used to perform the measurement are proven to be costly and time inefficient. Many breathing disorders can be detected through the measurement of the respiratory rate including sleep apnea and with early diagnosis to these disorders is crucial for treatment. There is an ever increasing demand in the medical for better ways to measure the respiratory rate of our patients and our proposed solution provides an efficient, simple, and low budget solution to measure respiratory rate. Through just an application installed to an android phone and a 400 dollars mobile thermal camera any user could measure his or her respiratory rate instantly.[29]

Moreover, due to the recent outbreak of the novel COVID-19 there is a demand for a simple, quick, accurate, and low cost solution for measuring the respiratory rate. COVID-19 effetely heavily the lungs of some patients diagnosed with COVID-19 and leads to difficulty in breathing. By measuring the respiratory rate we could identify quickly the patients that are suffering from breathing difficulties and provide them with ventilators. Our designed application should help with providing an easy and quick way to calculate the respiratory rate and it could improve the treatment process by early identification of patients with heavy COVID-19 symptoms. Thermal cameras are now almost spread everywhere to detect people with a high fever, and the technology and algorithm we use to calculate the respiratory rate through thermal image processing could be integrated to these thermal cameras to detect people that could be suffering from breathing abnormalities and possibly help with detecting people infected with COVID-19.[30]

2.1.1 Requirements

2.1.1.1 Accessibility requirements

The application is required to be accessible for everyone with an android smartphone as the aim of our project is to allow anyone to detect breathing disorders in a more simplified way than what is used in the real word. Therefore we intend to release our application on Google play store for anyone to download for free,

2.1.1.2 Functionality requirements

The application is required to detect any breathing disorder immediately. Therefore we had to design the app to perform all the processing on the phone itself instead of a server so that we avoid the delay that would come with using a server as we would need to send the images to the server for processing and then receiving the result back which would cause a small delay.

2.1.1.3 Usability requirements

We are designing the user interface of the application to be intuitive and easy to use, as the application is required to be user friendly and applicable for all people.

2.1.2 Constraints

2.1.2.1 Cost constraints

The budget of the project is required to not exceed 3500 dhs. Therefore we had to choose a mobile thermal camera with a reasonable price and a relatively cheap mobile phone for testing so that the total price won't go over the maximum budget. We had also chosen for our app to be designed as a native application and the image processing to be performed on the phone itself instead of a server to reduce the cost and also for functionality reasons.

2.1.2.2 Environment temperature constraint

The application can function properly only indoor. If the temperature of the room is too hot or too cold the temperature readings can be effected.The application should be designed to warn the user if the surrounding temperature is too hot or too cold.

2.1.2.3 User's distance from the phone constraints

The application is required to function accurately at a sensible range of distance from the phone for the user. The application should also inform the user of the distance range it can operate in. Test performed showed that the application can function accurately when the user's distance from the mobile device is between 50 cm and 10 cm.

2.1.3 Standards and codes considerations

The following standards were considered and incorporated in our design:

- Internet Access Management (Version 1.0), 2017, UAE Regulation Telecommunications Regulatory Authority.
- OMG Unified Modeling Language (Version 2.5), 2015, Object Management Group (OMG).
- IEEE 802.15.1-2005- IEEE Standard for Information technology -Local and metropolitan area networks- Specific requirements -Part 15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs), 2005, the IEEE.
- IEEE 802.11a -1999- Telecommunications and information exchange between systems - Local and metropolitan area networks— Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHZ Band, 1999, the IEEE.
- ISO/IEC/ IEEE 12207 -2017- Systems and software engineering - Software life cycle processes, 2017, the IEEE.
- ISO/IEC/ IEEE 29119-2-2013- Software and systems engineering - Software testing - Part 2: Test processes, 2013, the IEEE.
- IEEE 610.4 -1990- IEEE Standard Glossary of Image Processing and Pattern Recognition Terminology, 1990, the IEEE.
- IEC 60601 -2011- technical standards for the safety and essential performance of medical electrical equipment, 2011, International Electrotechnical Commission.
- MDD 93/42/EEC -2017- Standards for medical devices requirements, 2017, Medical Devices Directive.

We used the OMG Unified Modeling Language to visualize our application software design. The standards defined in the IEEE 610.4 -1990 was used in processing the thermal images continuously captured by our application. We ensured that the development, operation, and maintenance of the application software we designed were done in compliance to the ISO/IEC/ IEEE 12207 -2017 standards and we also confirmed that our project is done in accordance to the UAE's and Telecommunication Regulatory Authority Internet Access Management regulations. Also we used ISO/IEC/ IEEE 29119-2-2013 standards in performing the testing processes for our application software. For the wireless technology we intend to use, we considered Wi-Fi (over IEEE 802.15.1), Bluetooth (over IEEE 802.15.1), and ZigBee (over IEEE 802.15.4); eventually we decided on Wi-Fi technology being the most suitable technology. Finally, the application we designed, which is intended to be used as a medical device, was also designed in accordance to the IEC 60601 -2011- technical standards for the safety and essential performance of medical electrical equipment, and the requirements defined in the MDD 93/42/EEC -2017 standards was meet in our design.

2.2 System overview

Our proposed system utilizes a mobile thermal camera, specifically flir one mobile thermal camera, to measure breathing temperature continuously and detect from it the breath rate. The overall system consists of just an android mobile phone and the flir one mobile thermal camera as we described earlier. The system overview is shown in figure 2.1, and the only connection present in our system is the flir one mobile thermal camera connected to the mobile phone through a micro USB port. The application designed is an android native application designed through android studio and the application will use the flir one SDK to control the op2erations of the mobile thermal camera. The application will continuously monitor the users breath rate by measuring the thermal pixels of the pictures captured through the mobile thermal camera and plot them against time to form a breath rate signal.

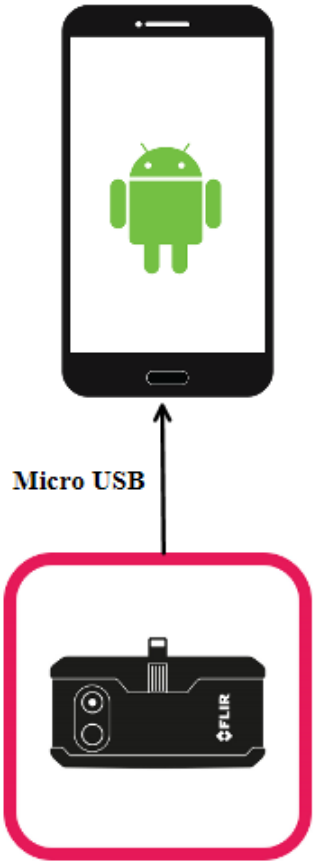

FIGURE 2.1: System overview

2.3 Component design

Our proposed system design consists of two components which are Flir one mobile thermal camera and android powered mobile device.

1) Flir one mobile thermal camera: It is a thermal imaging that renders infrared radiation to detect heated areas but for mobile phones. They can capture body heat and are useful for various other activities. Flir one also provide an SDK for android studio to control the operations of the camera. The flir one can capture images of different types including visible image and thermal kelvin radiated images. The flir one in our system will be used to capture images of both types where the visible images will be used to perform openCV processing operation on it to detect the area under the nose and then we will use the thermal image to measure the average temperature of the thermal pixels in the area detected through OpenCV. Flir one mobile thermal camera used is shown in

figure 1.18

2) Android mobile phone: The second component of our system is a mobile phone with Android OS. An application will be designed to run on this device. The application will designed through android studio as a native application where all the processing will occur on the phone itself, including image processing, and it will use the Flir one SDK to to utilize the thermal camera in capturing thermal and visible images, and measuring the temperature of thermal pixels. The application will also use an opneCV library to detect the area under the nose automatically. Additionally, the application will plot a graph for the breath rate and perform all the needed calculations for the respiratory rate on the device. An image of the android device used is shown in figure 2.2 below which is a Huawei nova 3i.

FIGURE 2.2: Android device used
[31]

The only connection present in our system is between the Flir one thermal camera and the android device through a micro USB port connection.

2.3.1 Implementation

2.3.1.1 Initial test

Our main objective of the initial tests performed is to proof the concept and to do so we need to ensure first that the mobile thermal camera we will use can provide continuously and accurate breath temperatures. The average human breath temperature is 34.5 degree Celsius and as we can see from figure 2.3 below, the Flir one mobile camera we used was able to provide us with very accurate and continuous measurements that are very close to the average human breath temperature with just a difference of 0.1 to 0.2 degree Celsius as the temperature measured in figure 2.3 during exhaling process is 34.7 degree Celsius.[32]

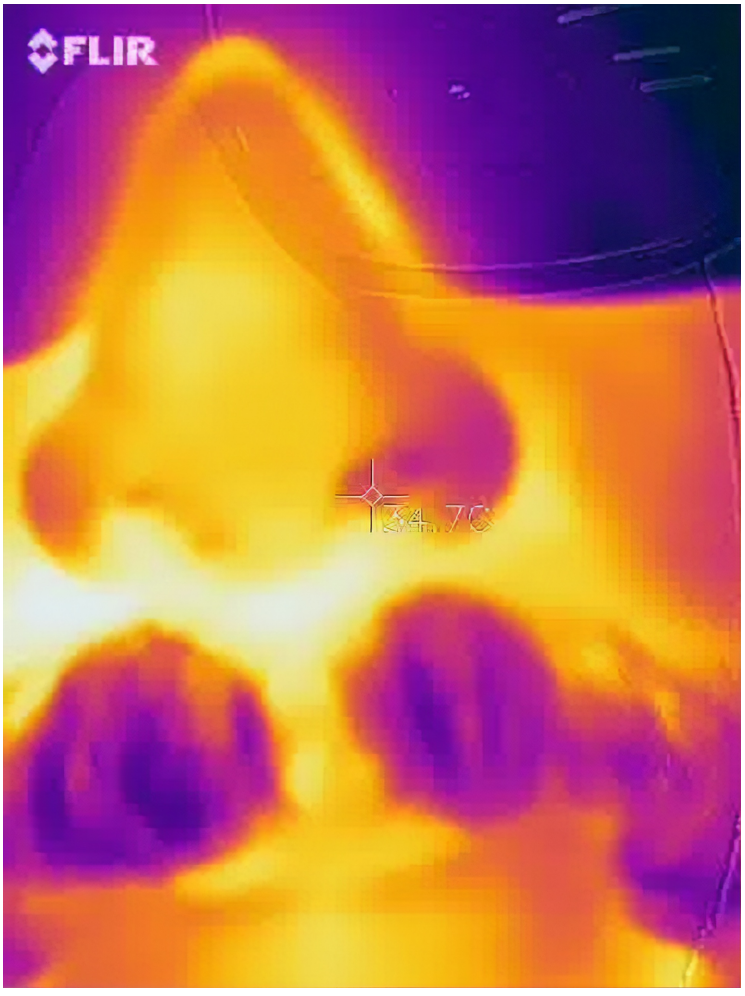

FIGURE 2.3: Exhaling thermal image

Figure 2.4 shows a breath measurement taken during the process of inhaling. We can see that the recorded temperature increases by a reasonable value, up to 35.9 degree Celsius, as expected. The difference can also be clearly seen from the thermal area highlighted in purple under the nostril that is present in the first figure 4 during exhaling but not in figure 2.4 for the inhaling process.[33]

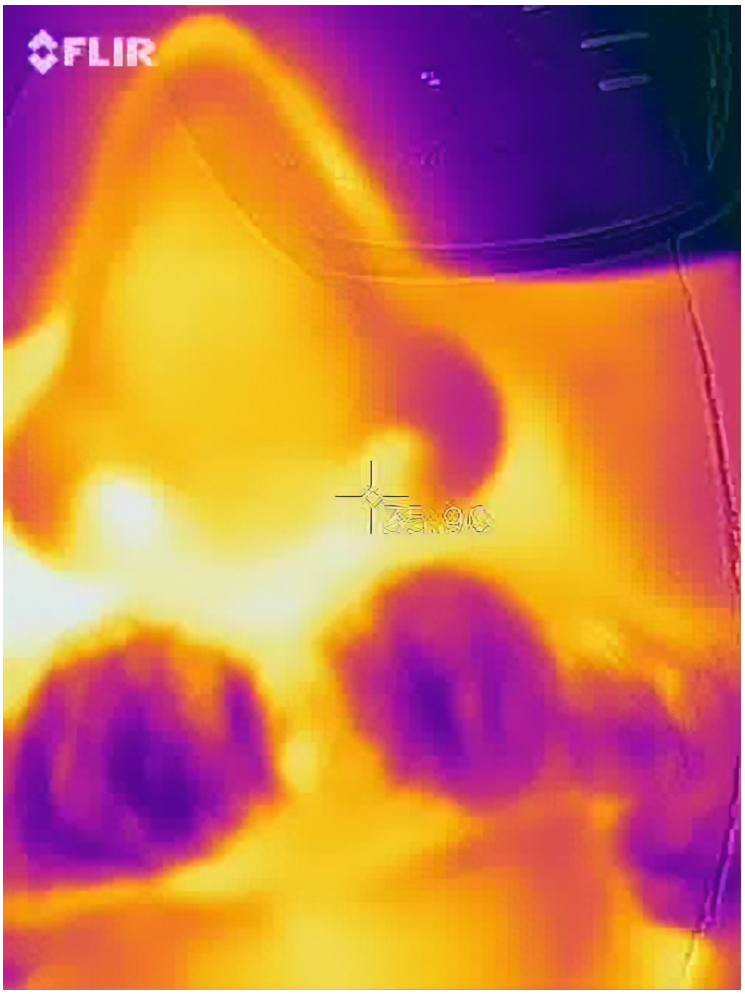

FIGURE 2.4: Inhaling thermal image

After we proofed the capability of the Flir one mobile thermal camera to capture accurate breath temperatures, we need to plot the recordings against time to form the signal. We took a 30 seconds video measuring continuously the breath temperature of a person and then split the video into frames using Matlab. The temperature measured at the first frame for each second is noted down manually and then sketched against time through matplotlib python library. The signal obtained is expected to be similar to a sine wave and as we can see from figure 2.5 below the signal obtained does actually look similar to a sine wave. The number of peaks represent the number of breaths taken by the user through that period of time. Here the signal formed in figure 2.5 has 7 peaks, which means that the user has taken 7 breaths per 30 seconds. If we extend the time interval to a minute the number of breaths taken by the user would be about 14 which is an accurate result as the average number of breaths per minute by an adult at reset is between 12 to 20 breaths.[34]

You might have noticed that the signal is sharper or steeper in comparison to a sine wave, that is because the values here are taken manually and just one value is taken per second, therefore if the value is taken automatically and continuously over time the results should be more similar to a sine-wave. Furthermore the positive cycle, in the signal formed, represent the inhaling process and the negative cycle represent the exhaling process. By obtaining an accurate signal of breath rate through thermal imaging, we have fully proofed the concept successfully.

2.3.1.2 Mobile application development

now in our mainly activity in GLPreviewActivity class we import the required libraries. Mainly the libraries in flir one sdk and opencv android sdk.

```
package com.flir.flirOneApplication;

import android.Manifest;
import android.annotation.TargetApi;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.content.res.Resources;
import android.graphics.Bitmap;
import android.graphics.Color;
import android.graphics.ColorFilter;
import android.opengl.GLSurfaceView;
import android.os.Build;
import android.os.Bundle;
```

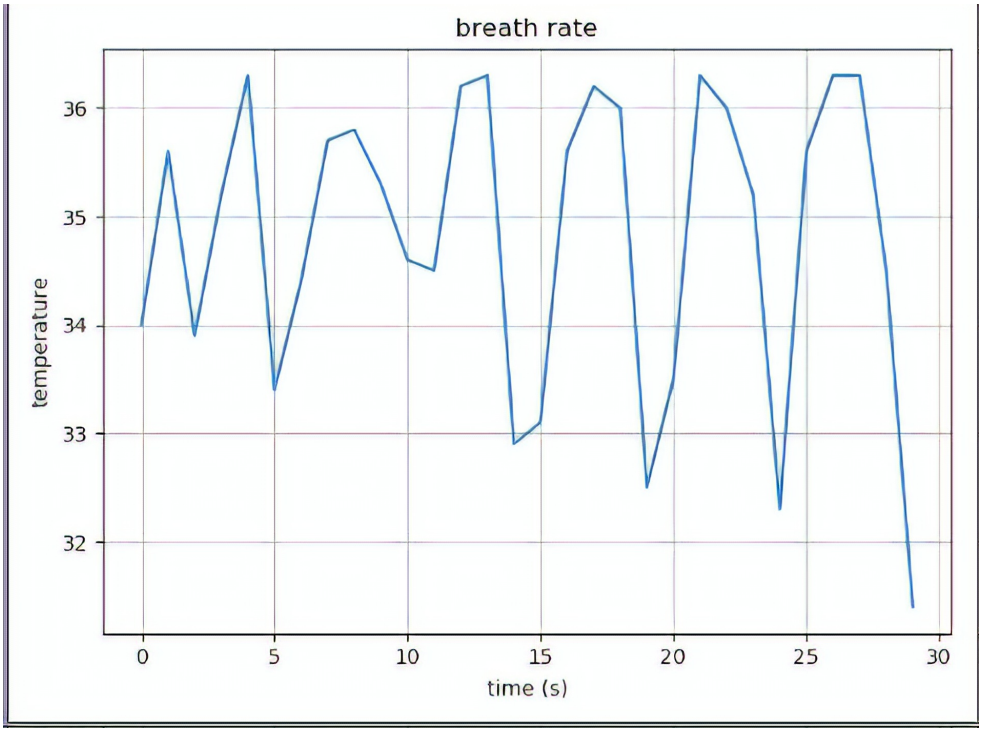

FIGURE 2.5: Initial tests Breath rate signal

```
import android.os.Handler;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v4.content.PermissionChecker;
import android.util.AttributeSet;
import android.util.Log;
import android.view.MotionEvent;
import android.view.OrientationEventListener;
import android.view.ScaleGestureDetector;
import android.view.SurfaceView;
import android.view.View;
import android.widget.AbsoluteLayout;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ToggleButton;

import com.flir.flironeexampleapplication.util.SystemUiHider;
import com.flir.flironesdk.Device;
import com.flir.flironesdk.FlirUsbDevice;
import com.flir.flironesdk.Frame;
import com.flir.flironesdk.FrameProcessor;
import com.flir.flironesdk.RenderedImage;
import com.flir.flironesdk.SimulatedDevice;
import com.jjoe64.graphview.series.DataPoint;
```

```
import org.opencv.android.BaseLoaderCallback;
import org.opencv.android.CameraBridgeViewBase;
import org.opencv.android.JavaCameraView;
import org.opencv.android.LoaderCallbackInterface;
import org.opencv.android.OpenCVLoader;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.core.MatOfByte;
import org.opencv.core.MatOfRect;
import org.opencv.core.Point;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;
import org.opencv.objdetect.CascadeClassifier;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.Socket;
import java.nio.ByteBuffer;
import java.text.NumberFormat;
import java.util.ArrayList;
import java.util.EnumSet;
import java.util.LinkedList;
import java.util.Map;
import java.util.Timer;
import java.util.TimerTask;
```

declare some variables we will use them in our implementation.

```
 LinearLayout linearLayout;
    CascadeClassifier faceDetector;
    private Mat mrgba , mGrey;
    File cascadeFile;


    GLSurfaceView thermalSurfaceView;
    private volatile boolean startRecording = false;
    double fractionOfSecond = 0 ;
    int seconds = 0;
    private volatile Socket streamSocket = null;
    private boolean chargeCableIsConnected = true;
    ArrayList<DataPoint> avrFrameTemp = new ArrayList<>();

    private int deviceRotation= 0;
    private OrientationEventListener orientationEventListener;


    private volatile Device flirOneDevice;
    private FrameProcessor frameProcessor;
```

```
private Device.TuningState currentTuningState = Device.TuningState.Unknown;
private boolean accFrame = true;


double rectX = 200;
double rectY = 400;
double rectHight = 1;
double rectWidth = 1;

double rectXpxl = 200;
double rectYpxl = 400;
double rectHightpxl = 1;
double rectWidthpxl = 1;
ArrayList<Frame> oldFrames = new ArrayList<Frame>();
ArrayList<Double> oldFramesTime = new ArrayList<>();

boolean addFrame = true;
boolean calculated = false;
boolean stopAcc = false;
```

First we start with implementing device delegate methods. Our first method is **onDeviceConnected** which Called during device discovery, when a device is connected. During this callback, we saved a reference to device then we also set the power update delegate for the device. We went ahead and started frame stream as soon as connected, in this use case. Finally we create a frame processor for rendering frames.

```
public void onDeviceConnected(Device device){
        Log.i("ExampleApp", "Device connected!");
        runOnUiThread(new Runnable() {
                          @Override
                          public void run() {
                              findViewById(R.id.pleaseConnect).setVisibility(View.GONE);
                          }
                      });

        flirOneDevice = device;
        flirOneDevice.setPowerUpdateDelegate(this);
        flirOneDevice.startFrameStream(this);

        final ToggleButton chargeCableButton = (ToggleButton)findViewById(R.id.chargeCableToggle
        if(flirOneDevice instanceof SimulatedDevice){
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    chargeCableButton.setChecked(chargeCableIsConnected);
                    chargeCableButton.setVisibility(View.VISIBLE);
                }
            });
        }else{
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    chargeCableButton.setChecked(chargeCableIsConnected);
```

```
                chargeCableButton.setVisibility(View.INVISIBLE);
                findViewById(R.id.connect_sim_button).setEnabled(false);

            }
        });
    }

    orientationEventListener.enable();
}
```

Then we implemented **onDeviceDisconnected** method to indicate to the user that the device has disconnected.

```
public void onDeviceDisconnected(Device device){

    final ToggleButton chargeCableButton = (ToggleButton)findViewById(R.id.chargeCableToggle
    final TextView levelTextView = (TextView)findViewById(R.id.batteryLevelTextView);
    final ImageView chargingIndicator = (ImageView)findViewById(R.id.batteryChargeIndicator)
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            findViewById(R.id.pleaseConnect).setVisibility(View.GONE);
            levelTextView.setText("--");
            chargeCableButton.setChecked(chargeCableIsConnected);
            chargeCableButton.setVisibility(View.INVISIBLE);
            chargingIndicator.setVisibility(View.GONE);
            findViewById(R.id.tuningProgressBar).setVisibility(View.GONE);
            findViewById(R.id.tuningTextView).setVisibility(View.GONE);
            findViewById(R.id.connect_sim_button).setEnabled(true);
        }
    });
    flirOneDevice = null;
    orientationEventListener.disable();
}
```

Because the thermal reading is not accurate during the tuned state . we implemented **onTuningStateChanged** to give us continuously update of the tuning state.

```
public void onTuningStateChanged(Device.TuningState tuningState){

    currentTuningState = tuningState;
    if (tuningState == Device.TuningState.InProgress){
        runOnUiThread(new Thread(){
            @Override
            public void run() {
                super.run();
                findViewById(R.id.tuningProgressBar).setVisibility(View.VISIBLE);
                findViewById(R.id.tuningTextView).setVisibility(View.VISIBLE);
            }
        });
    }else {
        runOnUiThread(new Thread() {
            @Override
            public void run() {
```

```
                super.run();
                findViewById(R.id.tuningProgressBar).setVisibility(View.GONE);
                findViewById(R.id.tuningTextView).setVisibility(View.GONE);
            }
        });
    }
}
```

we implemented two additional methods to show the user the pattery percentage.

```
public void onBatteryChargingStateReceived(final Device.BatteryChargingState batteryChargingS

    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            ImageView chargingIndicator = (ImageView)findViewById(R.id.batteryChargeIndicato
            if (originalChargingIndicatorColor == null){
                originalChargingIndicatorColor = chargingIndicator.getColorFilter();
            }
            switch (batteryChargingState) {
                case FAULT:
                case FAULT_HEAT:
                    chargingIndicator.setColorFilter(Color.RED);
                    chargingIndicator.setVisibility(View.VISIBLE);
                    break;
                case FAULT_BAD_CHARGER:
                    chargingIndicator.setColorFilter(Color.DKGRAY);
                    chargingIndicator.setVisibility(View.VISIBLE);
                case MANAGED_CHARGING:
                    chargingIndicator.setColorFilter(originalChargingIndicatorColor);
                    chargingIndicator.setVisibility(View.VISIBLE);
                    break;
                case NO_CHARGING:
                default:
                    chargingIndicator.setVisibility(View.GONE);
                    break;
            }
        }
    });
}
@Override
public void onBatteryPercentageReceived(final byte percentage){

    final TextView levelTextView = (TextView)findViewById(R.id.batteryLevelTextView);
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            levelTextView.setText(String.valueOf((int) percentage) + "%");
        }
    });


}
```

Here some additional options for the user. Where in these two methods if the user dose not have a device (mobile thermal camera) the software will simulate the code without the camera.

```
public void onConnectSimClicked(View v){
        if(flirOneDevice == null){
            try {
                flirOneDevice = new SimulatedDevice(this, this, getResources().openRawResource(R
                flirOneDevice.setPowerUpdateDelegate(this);
                chargeCableIsConnected = true;
            } catch(Exception ex) {
                flirOneDevice = null;
                Log.w("FLIROneExampleApp", "IO EXCEPTION");
                ex.printStackTrace();
            }
        }else if(flirOneDevice instanceof SimulatedDevice) {
            flirOneDevice.close();
            flirOneDevice = null;
        }
    }

public void onSimulatedChargeCableToggleClicked(View v){
        if(flirOneDevice instanceof SimulatedDevice){
            chargeCableIsConnected = !chargeCableIsConnected;
            ((SimulatedDevice)flirOneDevice).setChargeCableState(chargeCableIsConnected);
        }
}
```

Also there is an option for the user to capture rotated videos.

```
 public void onRotateClicked(View v){
        ToggleButton theSwitch = (ToggleButton)v;
        if (theSwitch.isChecked()){
            frameProcessor.setImageRotation(180.0f);
        }else{
            frameProcessor.setImageRotation(0.0f);
        }
    }
```

The user can change the view from thermal to any other view like visible or mix. That can be done through clicking on **onChangeViewClicked** and choose the required view from ImageTypeListView.

```
 public void onChangeViewClicked(View v){
        if (frameProcessor == null){
            ((ToggleButton)v).setChecked(false);
            return;
        }
        ListView paletteListView = (ListView)findViewById(R.id.paletteListView);
        ListView imageTypeListView = (ListView)findViewById(R.id.imageTypeListView);
        if (((ToggleButton)v).isChecked()){
            // only show palette list if selected image type is colorized
            paletteListView.setVisibility(View.INVISIBLE);
```

```
            for (RenderedImage.ImageType imageType : frameProcessor.getImageTypes()){
                if (imageType.isColorized()) {
                    paletteListView.setVisibility(View.VISIBLE);
                    break;
                }
            }
            imageTypeListView.setVisibility(View.VISIBLE);
            findViewById(R.id.imageTypeListContainer).setVisibility(View.VISIBLE);
        }else{
            findViewById(R.id.imageTypeListContainer).setVisibility(View.GONE);
        }


    }

    public void onImageTypeListViewClicked(View v){
        int index = ((ListView) v).getSelectedItemPosition();
        RenderedImage.ImageType imageType = RenderedImage.ImageType.values()[index];
        frameProcessor.setGLOutputMode(imageType);
        int paletteVisibility = (imageType.isColorized()) ? View.VISIBLE : View.GONE;
        findViewById(R.id.paletteListView).setVisibility(paletteVisibility);
    }
```

onStart method we searched about the connected device through the usb. There are may some exception could occur for example if we've already started discovery. Also on some platforms, we need the user to select the app to give us permission to the USB device.

```
protected void onStart(){
    super.onStart();
    mGrey = new Mat();
    mrgba = new Mat();
    if (Device.getSupportedDeviceClasses(this).contains(FlirUsbDevice.class)){
        findViewById(R.id.pleaseConnect).setVisibility(View.VISIBLE);
    }
    try {
        Device.startDiscovery(this, this);
    }catch(IllegalStateException e){

    }catch (SecurityException e){

        Toast.makeText(this,
        "Please insert FLIR One and select "+getString(R.string.app_name), Toast.LENGTH_LONG).sh

        finish();
    }
}
```

when the activity created the **onCreate** method will called. In this method we define some variables to meet the controls view. Then we set the default render image type to blended and create an object FrameProcessor which we will use in **onFrameProcessed**

method. We set Also the surface view to meet the GLSurfaceView from our design. Afterwords we added the image types and the ability to the user to change between them to change the view.Then Set up an instance of SystemUiHider to control the system UI for this activity.If the ViewPropertyAnimator API is available (Honeycomb MR2 and later), we will use it to animate the in-layout UI controls at the bottom of the screen. Upon interacting with UI controls, delay any scheduled hide() operations to prevent the jarring behavior of controls going away while interacting with the UI. In last part we are asking for permissions.

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

    setContentView(R.layout.activity_gl_preview);


    final View controlsView = findViewById(R.id.fullscreen_content_controls);
    final View controlsViewTop = findViewById(R.id.fullscreen_content_controls_top);
    final View contentView = findViewById(R.id.fullscreen_content);
    linearLayout = findViewById(R.id.fullscreen_content_controls_top);


    RenderedImage.ImageType defaultImageType = RenderedImage.ImageType.BlendedMSXRGBA8888Image;
    frameProcessor = new FrameProcessor(this, this, EnumSet.of(RenderedImage.ImageType.ThermalRa
    frameProcessor.setGLOutputMode(defaultImageType);


    thermalSurfaceView = (GLSurfaceView) findViewById(R.id.imageView);
    thermalSurfaceView.setPreserveEGLContextOnPause(true);
    thermalSurfaceView.setEGLContextClientVersion(2);
    thermalSurfaceView.setRenderer(frameProcessor);
    thermalSurfaceView.setRenderMode(GLSurfaceView.RENDERMODE_WHEN_DIRTY);
    thermalSurfaceView.setDebugFlags(GLSurfaceView.DEBUG_CHECK_GL_ERROR | GLSurfaceView.DEBUG_LO


    final String[] imageTypeNames = new String[]{ "Visible", "Thermal", "MSX" };
    final RenderedImage.ImageType[] imageTypeValues = new RenderedImage.ImageType[]{
            RenderedImage.ImageType.VisibleAlignedRGBA8888Image,
            RenderedImage.ImageType.ThermalRGBA8888Image,
            RenderedImage.ImageType.BlendedMSXRGBA8888Image,
    };

    ListView imageTypeListView = ((ListView)findViewById(R.id.imageTypeListView));
    imageTypeListView.setAdapter(new ArrayAdapter<>(this,R.layout.emptytextview,imageTypeNames))
    imageTypeListView.setSelection(defaultImageType.ordinal());
    imageTypeListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            if (frameProcessor != null) {
                RenderedImage.ImageType imageType = imageTypeValues[position];
                frameProcessor.setGLOutputMode(imageType);
                if (imageType.isColorized()){
```

```
                findViewById(R.id.paletteListView).setVisibility(View.VISIBLE);
            }else{
                findViewById(R.id.paletteListView).setVisibility(View.INVISIBLE);
            }
        }
    }
});
imageTypeListView.setDivider(null);

// Palette List View Setup
ListView paletteListView = ((ListView)findViewById(R.id.paletteListView));
paletteListView.setDivider(null);
paletteListView.setAdapter(new ArrayAdapter<>(this, R.layout.emptytextview, RenderedImage.Pa
paletteListView.setSelection(frameProcessor.getImagePalette().ordinal());
paletteListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        if (frameProcessor != null){
            frameProcessor.setImagePalette(RenderedImage.Palette.values()[position]);
        }
    }
});


mSystemUiHider = SystemUiHider.getInstance(this, contentView, HIDER_FLAGS);
mSystemUiHider.setup();
mSystemUiHider
        .setOnVisibilityChangeListener(new SystemUiHider.OnVisibilityChangeListener() {

            int mControlsHeight;
            int mShortAnimTime;

            @Override
            @TargetApi(Build.VERSION_CODES.HONEYCOMB_MR2)
            public void onVisibilityChange(boolean visible) {
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB_MR2) {

                    if (mControlsHeight == 0) {
                        mControlsHeight = controlsView.getHeight();
                    }
                    if (mShortAnimTime == 0) {
                        mShortAnimTime = getResources().getInteger(
                                android.R.integer.config_shortAnimTime);
                    }
                    controlsView.animate()
                            .translationY(visible ? 0 : mControlsHeight)
                            .setDuration(mShortAnimTime);
                    controlsViewTop.animate().translationY(visible ? 0 : -1 * mControlsHeigh
                } else {
                    // If the ViewPropertyAnimator APIs aren't
                    // available, simply show or hide the in-layout UI
                    // controls.
                    controlsView.setVisibility(visible ? View.VISIBLE : View.GONE);
                    controlsViewTop.setVisibility(visible ? View.VISIBLE : View.GONE);
                }
```

```
                if (visible && !((ToggleButton)findViewById(R.id.change_view_button)).isChec
                    // Schedule a hide().
                    delayedHide(AUTO_HIDE_DELAY_MILLIS);
                }
            }
        });

// Set up the user interaction to manually show or hide the system UI.
contentView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (TOGGLE_ON_CLICK) {
            mSystemUiHider.toggle();
        } else {
            mSystemUiHider.show();
        }
    }
});


findViewById(R.id.change_view_button).setOnTouchListener(mDelayHideTouchListener);


orientationEventListener = new OrientationEventListener(this) {
    @Override
    public void onOrientationChanged(int orientation) {
        deviceRotation = orientation;
    }
};
mScaleDetector = new ScaleGestureDetector(this, new ScaleGestureDetector.OnScaleGestureListe
    @Override
    public void onScaleEnd(ScaleGestureDetector detector) {
    }
    @Override
    public boolean onScaleBegin(ScaleGestureDetector detector) {
        return true;
    }
    @Override
    public boolean onScale(ScaleGestureDetector detector) {
        Log.d("ZOOM", "zoom ongoing, scale: " + detector.getScaleFactor());
        frameProcessor.setMSXDistance(detector.getScaleFactor());
        return false;
    }
});

findViewById(R.id.fullscreen_content).setOnTouchListener(new View.OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        mScaleDetector.onTouchEvent(event);
        return true;
    }
});
```

```
    String writePermission = Manifest.permission.WRITE_EXTERNAL_STORAGE;
    boolean permissionGranted = false;
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        permissionGranted = (ContextCompat.checkSelfPermission(this, writePermission) == Package
    }
    else {
        permissionGranted = (PermissionChecker.checkSelfPermission(this, writePermission) == Per
    }

    if(!permissionGranted) {
        if (ActivityCompat.shouldShowRequestPermissionRationale(this, writePermission)) {
            Toast.makeText(this, "App requires write permission to save photos", Toast.LENGTH_LO
        }
        else {
            ActivityCompat.requestPermissions(this, new String[]{writePermission}, 0);
        }
    }

    if(!OpenCVLoader.initDebug()){
        OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_4_0, this,baseCallback);

    }else{
        try {
            baseCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);

        } catch (IOException e) {

            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    ((TextView)findViewById(R.id.spotMeterValue))
                    .setText("manager dose not connected ");
                }
            });

        }

    }


}
```

Here the implementation of onPause, onResume and onStop methods.

```
public void onPause(){
    super.onPause();

    thermalSurfaceView.onPause();
    if (flirOneDevice != null){
        flirOneDevice.stopFrameStream();
    }
}
@Override
public void onResume(){
    super.onResume();
```

```
    thermalSurfaceView.onResume();

    if (flirOneDevice != null){
        flirOneDevice.startFrameStream(this);
    }
}
@Override
public void onStop() {

    mrgba.release();
    mGrey.release();
    Device.stopDiscovery();
    flirOneDevice = null;
    super.onStop();
}
```

now we initialize the functionality of openCV. when the loader callback interface succeeded we got the classifier from the place where the haarcascade-mcs classifier stored in the project. Then we wrote this classifier in a file in the Mobile to re use it again in the future. Finally, we define the nose Detector as new cascadeClassifier from the cascade file.

```
private BaseLoaderCallback baseCallback = new BaseLoaderCallback(this) {
    @Override
    public void onManagerConnected(int status) throws IOException {

        switch (status){
            case LoaderCallbackInterface.SUCCESS:{


                InputStream is = getResources().openRawResource(R.raw.haarcascade_mcs_nose);
                File cascadeDir = getDir("cascade" , Context.MODE_PRIVATE);
                cascadeFile = new File(cascadeDir , "haarcascade_mcs_nose.xml");

                FileOutputStream fos = new FileOutputStream(cascadeFile);
                byte[] buffer = new byte[4096];
                int bytesRead;

                while ((bytesRead = is.read(buffer)) != -1){
                    fos.write(buffer , 0 , bytesRead);
                }
                is.close();
                fos.close();
               ;
                noseDetector = new CascadeClassifier((cascadeFile.getAbsolutePath()));
                if(faceDetector.empty()){
                    faceDetector=null;
                }else
                    cascadeDir.delete();
```

```
            }break;

            default:
            {
                super.onManagerConnected(status);
            }


        }

    }
};
```

In the **onFrameReceived** and **onFrameProcessed** methods we have three approaches to implement them

2.3.1.3 approach 1

In approach 1 we tried to measure the breath rate without using openCV. We drew a fixed rectangle the user has to fix the area under his nose in this rectangle and start recording. We measure the average temperature in this rectangle in each frame received during the recording time. First in **onFrameReceived** we processed each frame received if the device is not tuning state. Then we request render the image.

In **onFrameProcessed** we received only one image type ThermalRadiometricKelvin-Image . We found the pixel which represent our rectangle. After that we got the temperature from each pixel in the rectangle and then we calculated the average temperature. Finally, we append the average temperature from each frame with the time it captured at.

```
public void onFrameReceived(Frame frame) {
    if (currentTuningState != Device.TuningState.InProgress){
        frameProcessor.processFrame(frame, FrameProcessor.QueuingOption.CLEAR_QUEUED);
        thermalSurfaceView.requestRender();
    }
}

private Bitmap thermalBitmap = null;


public void onFrameProcessed(final RenderedImage renderedImage){

    if ( startRecording && renderedImage.imageType() == RenderedImage.ImageType.ThermalRadiometr

        int[] thermalPixels = renderedImage.thermalPixelValues();

        int width = renderedImage.width();
        int height = renderedImage.height();
        int startPixelIndex = (int)(((width) * (height*0.45)) +(width*0.3));
```

```
        int indexexNo = (int)((width*0.3) *(height*0.1));
        int[] centerPixelIndexes = new int[indexexNo];
        int newWidth = (int) (width*0.3);

        for(int x = 0 ; x < centerPixelIndexes.length ; x++){
            centerPixelIndexes[x] = startPixelIndex +x;

            if (x %newWidth == 0 && x !=0){
                startPixelIndex += (int)(width - (width*0.3));
            }

        }

        double averageTemp = 0;

        for (int i = 0; i < centerPixelIndexes.length; i++){
            int pixelValue = (thermalPixels[centerPixelIndexes[i]]);

            averageTemp += (((double)pixelValue) - averageTemp) / ((double) i + 1);
        }
        double averageC = (averageTemp / 100) - 273.15;
        double second = fractionOfSecond/10.0;
        DataPoint tempPoint = new DataPoint(second , averageC);
        avrFrameTemp.add(tempPoint);

        NumberFormat numberFormat = NumberFormat.getInstance();
        numberFormat.setMaximumFractionDigits(2);
        numberFormat.setMinimumFractionDigits(2);

        String spotMeterValue = numberFormat.format(averageC) + " C ";
        TextView textView = ((TextView)findViewById(R.id.spotMeterValue));
        textView.setText(spotMeterValue);

    }




}
```

2.3.1.4 approach 2

In this approach we use the openCV to detect the area under the nose and keep tracking in while recording. Here in **onFrameReceived** we processed each frame received if the device is not in tuning state and we added a new condition which is when the previous frame is processed successfully. Then we request render the image. In **onFrameProcessed** we received two image type ThermalRadiometricKelvinImage and VisibleAlignedRGBA8888Image .We need the visible image for open cv process. In openCV processing we need to convert the rendered image to Mat which is the main

matrix class in openCV. OpenCV by using the haarcascade classifier will create a rectangle for the nose in the image. We made some calculation to draw a rectangle under the nose depend on the rectangle of openCV and we found the pixel which represent our rectangle. After that we got the temperature from each pixel in the rectangle and then we calculated the average temperature.OpenCV will keep tracking the required area in each frame. Finally, we append the average temperature from each frame with the time it captured at.

```
public void onFrameReceived(Frame frame) {

    if (currentTuningState != Device.TuningState.InProgress && accessFrame){

        frameProcessor.processFrame(frame, FrameProcessor.QueuingOption.CLEAR_QUEUED);
        thermalSurfaceView.requestRender();
    }


}

private Bitmap thermalBitmap = null;

public void onFrameProcessed(final RenderedImage renderedImage){
    try {

       if(renderedImage.imageType() == RenderedImage.ImageType.VisibleAlignedRGBA8888Image  ){
            if(startRecording)
                stopAcc= true;
            accFrame = false;
            mrgba = new Mat(renderedImage.width() , renderedImage.height(), CvType.CV_8UC3);
            //mrgba =  Imgcodecs.imdecode(new MatOfByte(renderedImage.pixelData()), Imgcodecs.CV
            mrgba.put(0,0, renderedImage.pixelData());
            final MatOfRect noseDetection = new MatOfRect();
            noseDetector.detectMultiScale(mrgba , noseDetection);

            double width = renderedImage.width();
            double height = renderedImage.height();
            double screenX = Resources.getSystem().getDisplayMetrics().widthPixels;
            double screenY = Resources.getSystem().getDisplayMetrics().heightPixels;

            double opencvWidth = mrgba.width();
            double opencvHeight = mrgba.height();


            for (Rect rect:faceDetection.toArray()){

                rectX = (rect.x)  * (screenX/opencvWidth) ;
                rectXpxl = rect.x  ;
                rectY = (rect.y + (rect.height/2.0)) * (screenY/opencvHeight) ;
                rectYpxl = rect.y + (rect.y/2.0) ;
                rectHight = (rect.height /2.0) * (screenX/opencvWidth);
                rectHightpxl = rect.height/2.0;
                rectWidth = (rect.width ) *(screenX/opencvWidth);
```

```
            rectWidthpxl = (rect.width);
        }


        /*
        TextView T1 = ((TextView)findViewById(R.id.spotMeterValue));
        String bla = opencvWidth + " " + opencvHeight ;
        T1.setText(bla);*/

       AbsoluteLayout rect = (AbsoluteLayout)findViewById(R.id.recDetect);
         rect.setX((int)rectX);
         rect.setY((int)rectY);
         rect.getLayoutParams().height = (int)rectHight;
         rect.getLayoutParams().width = (int)rectWidth;

         if(faceDetection.toArray().length == 0){
             rect.setX(0);
             rect.setY(0);
             rect.getLayoutParams().height = 0;
             rect.getLayoutParams().width = 0;
             /*
             rectXpxl = 0 ;
             rectYpxl = 0;
             rectHightpxl = 0;
             rectWidthpxl = 0;*/
         }


       accFrame = true;
   }
 }catch (final Exception ex ){
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            ((TextView)findViewById(R.id.spotMeterValue)).setText("error "+ex.getMessage());
        }
    });
}


if (renderedImage.imageType() == RenderedImage.ImageType.ThermalRadiometricKelvinImage && st

    int[] thermalPixels = renderedImage.thermalPixelValues();

    int width = renderedImage.width();
    int height = renderedImage.height();
    int startPixelIndex = (int)(((rectYpxl-1) * (rectWidthpxl)) + rectXpxl);
    int indexexNo = (int)(rectWidthpxl*rectHightpxl);
    int[] centerPixelIndexes = new int[indexexNo];
    int newWidth = (int) rectWidthpxl;

    for(int x = 0 ; x < centerPixelIndexes.length ; x++){
        centerPixelIndexes[x] = startPixelIndex +x;

        if (x %newWidth == 0 && x !=0){
```

```
                startPixelIndex += (int)(width - rectWidthpxl);
            }

        }

        double averageTemp = 0;

        for (int i = 0; i < centerPixelIndexes.length; i++){
            // Remember: all primitives are signed, we want the unsigned value,
            // we've used renderedImage.thermalPixelValues() to get unsigned values

            int pixelValue = (thermalPixels[centerPixelIndexes[i]]);

            averageTemp += (((double)pixelValue) - averageTemp) / ((double) i + 1);
        }
        double averageC = (averageTemp / 100) - 273.15;
        double second = fractionOfSecond/10.0;
        DataPoint tempPoint = new DataPoint(second , averageC);
        avrFrameTemp.add(tempPoint);

        NumberFormat numberFormat = NumberFormat.getInstance();
        numberFormat.setMaximumFractionDigits(2);
        numberFormat.setMinimumFractionDigits(2);

        String spotMeterValue = numberFormat.format(averageC) +" C ";
        TextView textView = ((TextView)findViewById(R.id.spotMeterValue));
        textView.setText(spotMeterValue);
        stopAcc = false;
    }




}
```

2.3.1.5 approach 3

In approach 3 we tried to reduce the number of openCV operations by making it detect the required area every 5 seconds during recording the average temperature. By reducing the number of operations done by openCV we got better signal.

```
public void onFrameReceived(Frame frame) {



    if (currentTuningState != Device.TuningState.InProgress){


        if (accFrame ) {
            frameProcessor.processFrame(frame, FrameProcessor.QueuingOption.CLEAR_QUEUED);
```

```
            thermalSurfaceView.requestRender();
        }



    }







}

private Bitmap thermalBitmap = null;

// Frame Processor Delegate method, will be called each time a rendered frame is produced
public void onFrameProcessed(final RenderedImage renderedImage){
    try {

        if(renderedImage.imageType() == RenderedImage.ImageType.VisibleAlignedRGBA8888Image && a
            if(startRecording)
                stopAcc= true;
            accFrame = false;
            mrgba = new Mat(renderedImage.width() , renderedImage.height(), CvType.CV_8UC3);
            //mrgba =  Imgcodecs.imdecode(new MatOfByte(renderedImage.pixelData()), Imgcodecs.CV
            mrgba.put(0,0, renderedImage.pixelData());
            final MatOfRect faceDetection = new MatOfRect();
            faceDetector.detectMultiScale(mrgba , faceDetection);

            double width = renderedImage.width();
            double height = renderedImage.height();
            double screenX = Resources.getSystem().getDisplayMetrics().widthPixels;
            double screenY = Resources.getSystem().getDisplayMetrics().heightPixels;

            double opencvWidth = mrgba.width();
            double opencvHeight = mrgba.height();


            for (Rect rect:faceDetection.toArray()){

                rectX = (rect.x +(rect.x * (1/5.0)))  * (screenX/opencvWidth) ;
                rectXpxl = rect.y +(rect.x * (2/5.0));
                rectY = (rect.y + (rect.height/2.0)) * (screenY/opencvHeight) ;
                rectYpxl = rect.x + (rect.width/2.0);
                rectHight = (rect.height /2.0) * (screenX/opencvWidth);
                rectHightpxl = rect.width*0.6;
                rectWidth = (rect.width * 0.8) *(screenX/opencvWidth);
                rectWidthpxl = rect.height* 0.8 ;
            }



            AbsoluteLayout rect = (AbsoluteLayout)findViewById(R.id.recDetect);
```

```
            rect.setX((int)rectX);
            rect.setY((int)rectY);
            rect.getLayoutParams().height = (int)rectHight;
            rect.getLayoutParams().width = (int)rectWidth;

            if(faceDetection.toArray().length == 0){
                rect.setX(0);
                rect.setY(0);
                rect.getLayoutParams().height = 0;
                rect.getLayoutParams().width = 0;

                 rectXpxl = 0 ;
                 rectYpxl = 0;
                 rectHightpxl = 0;
                 rectWidthpxl = 0;
            }


            accFrame = true;
        }
    }catch (final Exception ex ){
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                ((TextView)findViewById(R.id.spotMeterValue)).setText("error "+ex.getMessage());
            }
        });
    }


    if (renderedImage.imageType() == RenderedImage.ImageType.ThermalRadiometricKelvinImage && st
        // Note: this code is not optimized

        int[] thermalPixels = renderedImage.thermalPixelValues();
        // average the center 9 pixels for the spot meter

        int theWidth = renderedImage.width();

        int startPixelIndex = (int)(((rectYpxl-1) * (rectWidthpxl)) + rectXpxl);
        int indexexNo = (int)(rectWidthpxl*rectHightpxl);
        int[] centerPixelIndexes = new int[indexexNo];
        int newWidth = (int) rectWidthpxl;

        for(int x = 0 ; x < centerPixelIndexes.length ; x++){
            centerPixelIndexes[x] = startPixelIndex +x;

            if (x %newWidth == 0 && x !=0){
                startPixelIndex += (int)(theWidth - rectWidthpxl);
            }

        }

        double averageTemp = 0;

        for (int i = 0; i < centerPixelIndexes.length; i++){
```

```
            // Remember: all primitives are signed, we want the unsigned value,
            // we've used renderedImage.thermalPixelValues() to get unsigned values

            int pixelValue = (thermalPixels[centerPixelIndexes[i]]);

            averageTemp += (((double)pixelValue) - averageTemp) / ((double) i + 1);
        }
        double averageC = (averageTemp / 100) - 273.15;
        double second = fractionOfSecond/10.0;
        DataPoint tempPoint = new DataPoint(second , averageC);
        avrFrameTemp.add(tempPoint);

        NumberFormat numberFormat = NumberFormat.getInstance();
        numberFormat.setMaximumFractionDigits(2);
        numberFormat.setMinimumFractionDigits(2);

        String spotMeterValue = numberFormat.format(averageC) + " C ";
        TextView textView = ((TextView)findViewById(R.id.spotMeterValue));
        textView.setText(spotMeterValue);
        stopAcc= false;
    }




}



public void onCaptureImageClicked(View v){
    accessOpenCV = false;
    if(flirOneDevice != null) {
        this.startRecording = true;
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                ((ImageButton)findViewById(R.id.imageButton)).setEnabled(false);
            }
        });

        new Timer().scheduleAtFixedRate(new TimerTask(){
            @Override
            public void run(){
                fractionOfSecond++;
                if(fractionOfSecond %10 ==0){
                    seconds++;

                    if(seconds %5 == 0)
                        accessOpenCV = true;
                    else
                        accessOpenCV = false;

                    if(seconds<10) {
                        runOnUiThread(new Runnable() {
```

```
                @Override
                public void run() {
                    ((TextView) findViewById(R.id.SecondsPreview)).setText("0:0" + s
                }
            });
        }
        else {
            runOnUiThread(new Runnable() {

                @Override
                public void run() {
                    ((TextView) findViewById(R.id.SecondsPreview)).setText("0:" + se
                }
            });
        }

    }

    if(seconds == 30){
        //addFrame = false;
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                ((ImageButton)findViewById(R.id.imageButton)).setEnabled(true);
            }
        });


        seconds = 0 ;
        fractionOfSecond =0;
        startRecording = false;
        calculated = true;

        try {
            /*
            if(accFrame){
                Frame nframe = oldFrames.remove(0);
                cTime = oldFramesTime.remove(0);
                frameProcessor.processFrame(nframe, FrameProcessor.QueuingOption.CLE
                thermalSurfaceView.requestRender();
            }*/
            calculateBreathRate();
            this.cancel();
            /*
            if(oldFrames.size() == 0){

            }*/

        }catch (final Exception ex){
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    ((TextView)findViewById(R.id.spotMeterValue)).setText("error " +
                }
```

```
                        });
                    }

                }
            }
        },0,100);
    }
}
```

In all approaches we need to draw the signal and calculate number of peaks.We drew the signal through GraphView class where we added a series of collected datapoint represent the average temperature and corresponding time. Then by made for loop across all points to find where the signal increasing then decreasing to find the peaks. Our calculation is for 30 seconds, so finally we multiplied the result by 2 to get the respiratory rate in minute.

```
import android.media.Image;
import android.support.annotation.DrawableRes;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ImageView;
import android.widget.TextView;

import com.jjoe64.graphview.GraphView;
import com.jjoe64.graphview.series.DataPoint;
import com.jjoe64.graphview.series.LineGraphSeries;

import java.util.ArrayList;

public class resultsPreview extends AppCompatActivity {
    private ArrayList<DataPoint> rsltArr ;
    LineGraphSeries<DataPoint> series;
    int peakCount = 0;

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

        rsltArr = (ArrayList<DataPoint>) getIntent().getSerializableExtra("resultsList");

        double x, y ;
        x =0 ;

        GraphView graph = (GraphView)findViewById(R.id.signalView);
        ImageView tickView = (ImageView)findViewById(R.id.tickView);

        series = new LineGraphSeries<DataPoint>();

        series.setTitle("Breath rate");

        double nextNextTemp, nextTemp, currentTemp;
```

```
ArrayList path = new ArrayList();
String prevState = "";
String currentState = "" ;


final double lastTime = rsltArr.get(rsltArr.size()-1).getX();
for (int i =0 ; i< rsltArr.size()-2; i++){
    series.appendData(rsltArr.get(i), false, 270);
    currentTemp = rsltArr.get(i).getY();
    nextTemp = rsltArr.get(i+1).getY();
    nextNextTemp = rsltArr.get(i+2).getY();
    if(currentTemp < nextTemp && nextTemp >nextNextTemp){
        peakCount++;
    }

}

graph.addSeries(series);

graph.setTitle("Breath rate signal");
graph.setTitleTextSize(72);

graph.setPadding(5, 5, 5, 5);
graph.getViewport().setScalable(true);

runOnUiThread(new Runnable() {
    @Override
    public void run() {
        ((TextView)findViewById(R.id.breathRateView)).setText("Breath rate = " + (peakCo
    }
});

if((peakCount*2)>25||(peakCount*2)<15) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            ((TextView) findViewById(R.id.resultReview)).setText("Your breathing is abno
        }
    });

    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            ((ImageView) findViewById(R.id.tickView)).setImageResource(R.drawable.ic_war
        }
    });

}

else {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            ((TextView) findViewById(R.id.resultReview)).setText("Your breathing is norm
        }
```

```
            });

            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    ((ImageView) findViewById(R.id.tickView)).setImageResource(R.drawable.ic_che
                }
            });

        }


    }
}
```

2.3.1.6 Application testing plan

As we have discussed before there are 3 approaches that we considered in designing our application which are monitoring the breath temperature for a constant area decided manually for the user to input the area under his nose in, or continuously detecting the area under the nose automatically while recording the temperature, or detecting the area under the nose one time before starting to record and keep measuring the temperature of that area continuously. We have conducted three test to determine which approach to use and how accurate each approach is.

- First we tested the result accuracy of each approach where the user is close from the phone and facing the phone from a straight angle. The acceptance criteria of this test is is for the application to provide a smooth signal that is shaped like a sine wave and the breath rate calculated is in line with the signal produced.
- After we have conducted the first test for the 3 approaches we will perform a second test, for the approaches that passed the first, on the maximum distance, from the user's face, the application can operate accurately. We will test each approach on different distances until we reach a distance were the application delivers inaccurate results to determine the maximum distance were the application can deliver accurate results.
- Third and final test we will conduct is for the maximum angle deviation that the application can accept to still deliver accurate result. We will test the application on different angles to see the maximum angle deviation that application can operate on.

2.4 Flow chart

Figure 2.6 shows a simple flow chart that describes the device functionality and the overall objective of the device. As we can see from the chart will wait for the user to press the start recording button. Once it is pressed the device will detect the area under the nose of the patient, from a visible image captured through the mobile thermal camera, through OpenCV. Afterwards, it will measure the average temperature of that area detected in degrees. The device will continue to do that for 30 seconds and afterwards it will display the breath rate signal and the value calculated for the breath rate. Finally it will display to the user if the respiratory rate calculated for the patient is abnormal or not.

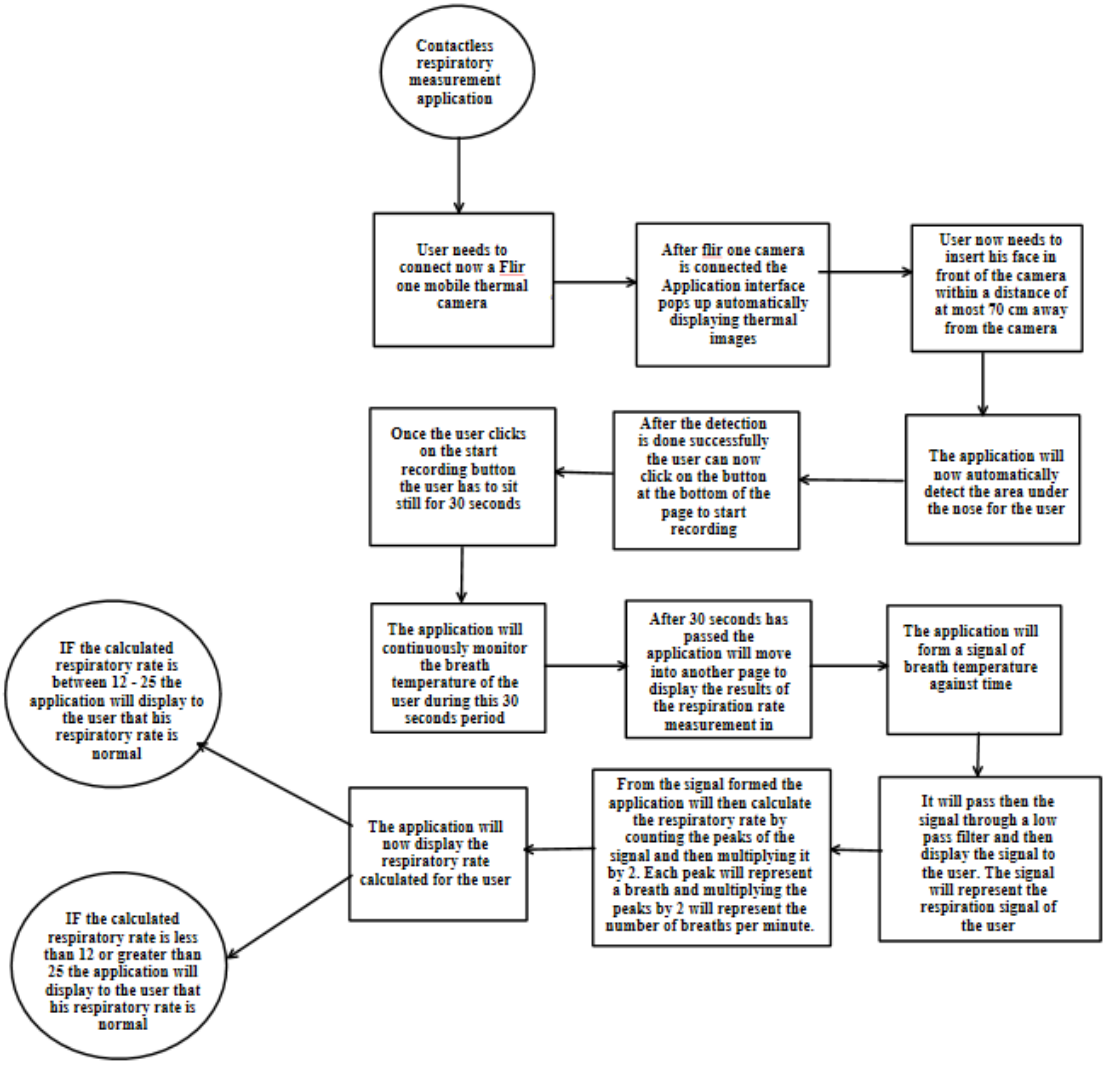

FIGURE 2.6: System flow chart

2.5 Design process

The design process for our application was based on the design constraints and requirements we mention in section **??** and in line with the standards and codes mentioned in 2.1.3. The design process of our application is divided into 3 main stages where first we have to install flir one SDK and openCV into out android IDE. Then we designed the application main interface and then lastly, the results page interface.

2.5.1 Installation procedure

2.5.1.1 Flir one SDk Installation

So first things first, to start designing our application on android studio we need to install the flir one SDK and add its dependencies. To install the flir one SDk for android studio you need to visit the website https://developer.flir.com/mobile/flironesdk/ and download the flir one SDk for android studio. The download will provide with a zip that not just containts the flir one SDk, it also containts documentation for the SDK and an example android project that uses flir one SDK. We heavily recommend to read the documentation and exlpore the example application to understand the different class and methods flir one library provides. Once the download is done you need to integrate the flir one SDk into your android project by following the project setup guide at https://developer.flir.com/getting-started/android-platform-guide-flir-one-cat-s60/ and then you are good to start designing your flir one android application.[35]

2.5.1.2 OpenCV Installation

To install openCV we need to download openCV android sdk.Then we have to import openCV module from file to new then import module then from opencv android sdk choose java. Now we need to link to our project we can do that by going to file then project structure then dependencies then click on add operator then module dependency then openCV as shown in figure 2.7 below.

We also need to add permission to use the camera in the application to make openCV work. Additionally, we have to copy all opencv sdk library in jinLibs folder in the project. finally, we added haarcascade classifier to detect the nose. [36]

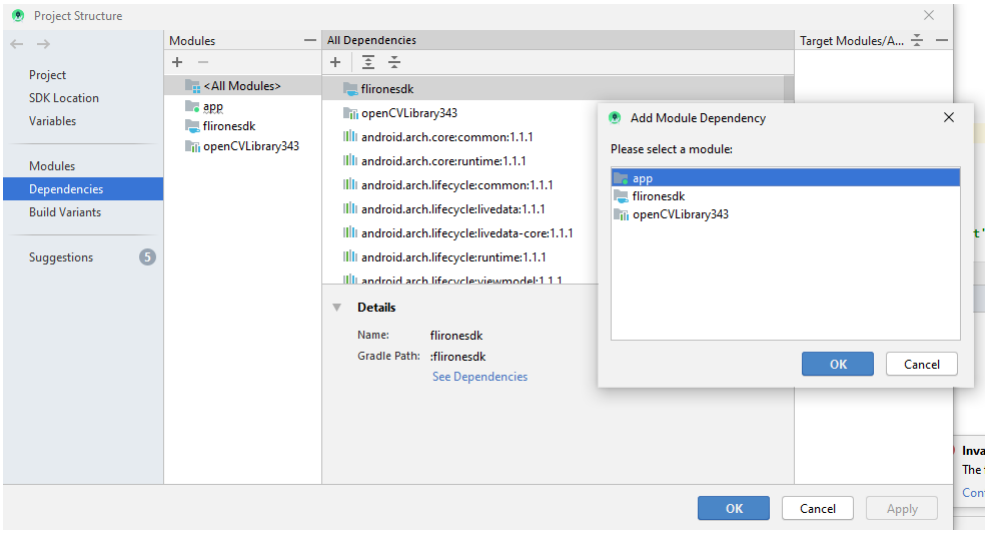

FIGURE 2.7: Import OpenCV module

2.5.2 Application interface design

As we have stated multiple times before the application should be user friendly and simple to use, and to achieve that it will all depend on the applications interface. Figure 2.8 below shows the applications API at the start. Once the user clicks on the application and has his flir one mobile thermal camera connected, it will show a surface view that displays the frames captured by the thermal camera. The API also contains a button on top called tune to tune and re-calibrate the camera when needed and it is recommended to tune the camera from time to time to obtain more accurate results and it also displays the flir one camera battery percentage for the user on the top left of the page.

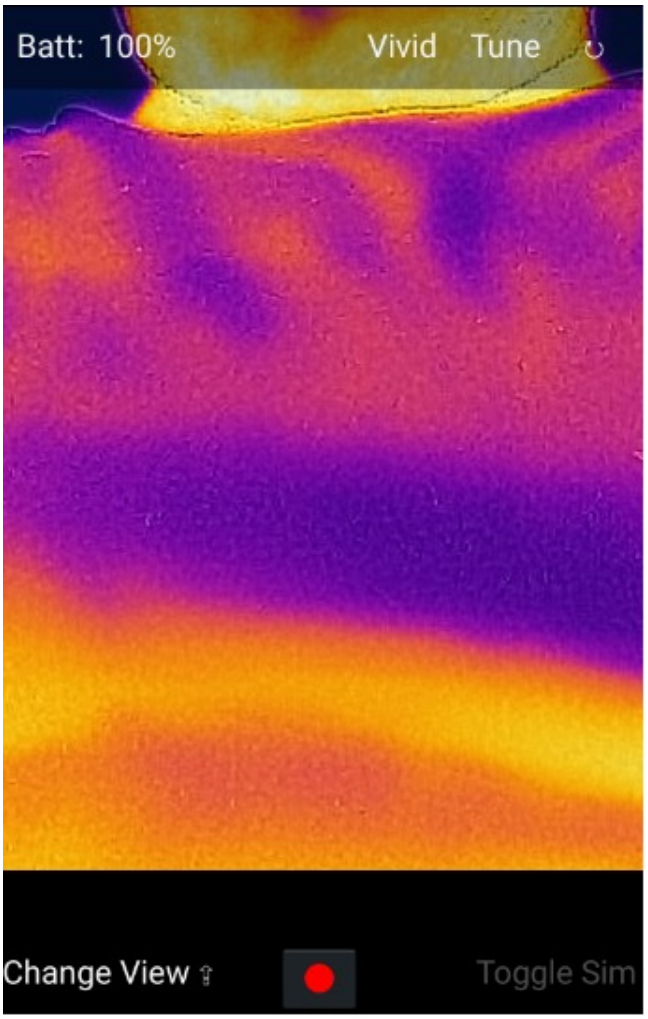

FIGURE 2.8: Application interface

On the interface, a rectangle will also be placed on the area under the user's nose if a face is detected as we can see from the figure 2.9 below.

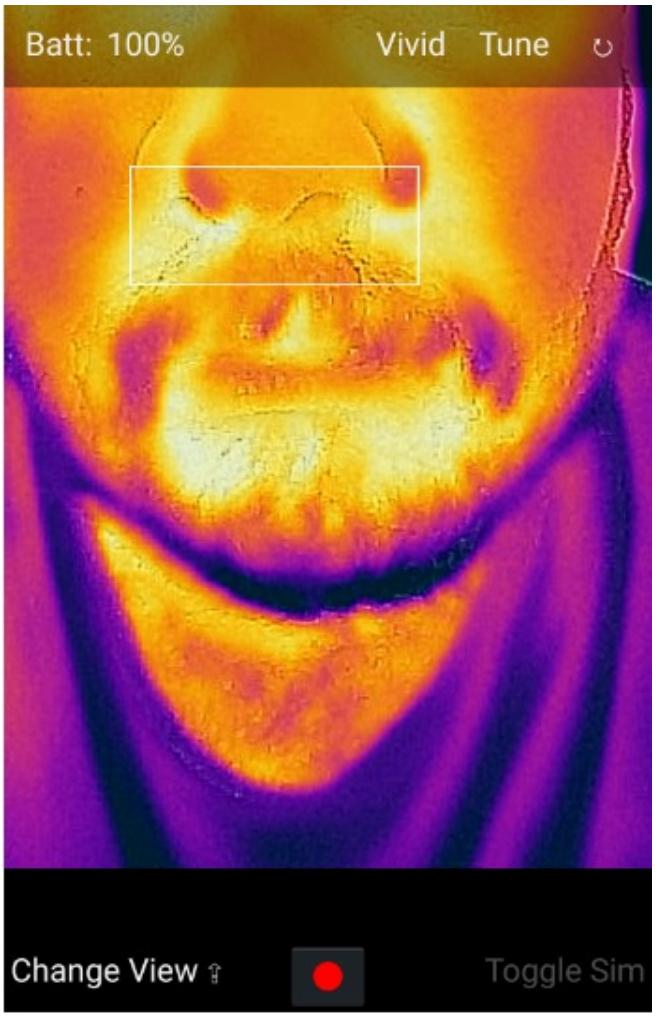

FIGURE 2.9: Area under the nose detection

There is also a button on the bottom to start recording and once it is pressed the button will be disabled and a timer will start on the button screen and the temperature being recorded for the area detected is displayed on the screen, in real time, as displayed in figure 2.10 below. Once the timer reaches 30 seconds the android application will switch to an activity that displays the results.

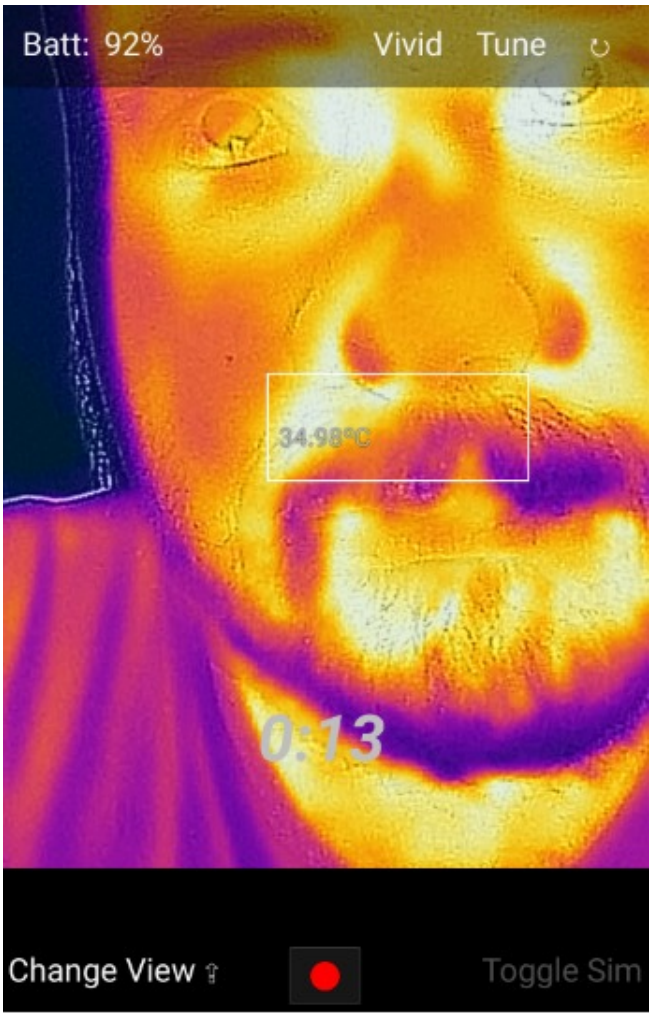

FIGURE 2.10: API after starting to record

2.5.3 Results page interface design

After the application is done from recording the user's breath temperature for 30 seconds it will move into a page to display the result of the test. The results API is displayed in figure 2.11. As you can see the page is titled results and on the top half of the page a graph view is present to display the breath temperature measured against time and form a breath rate signal. On the bottom half of the page it will show to the user the calculated respiratory rate and inform the user if the calculated breath rate is abnormal or normal through a text view and an image beside the calculated breath rate that shows a tick if it is normal and alert sign if it is abnormal, to make the application more user friendly and easier to understand for the user. Figure 2.11 is for a result page for a user with normal respiratory rate; between 15 - 25.

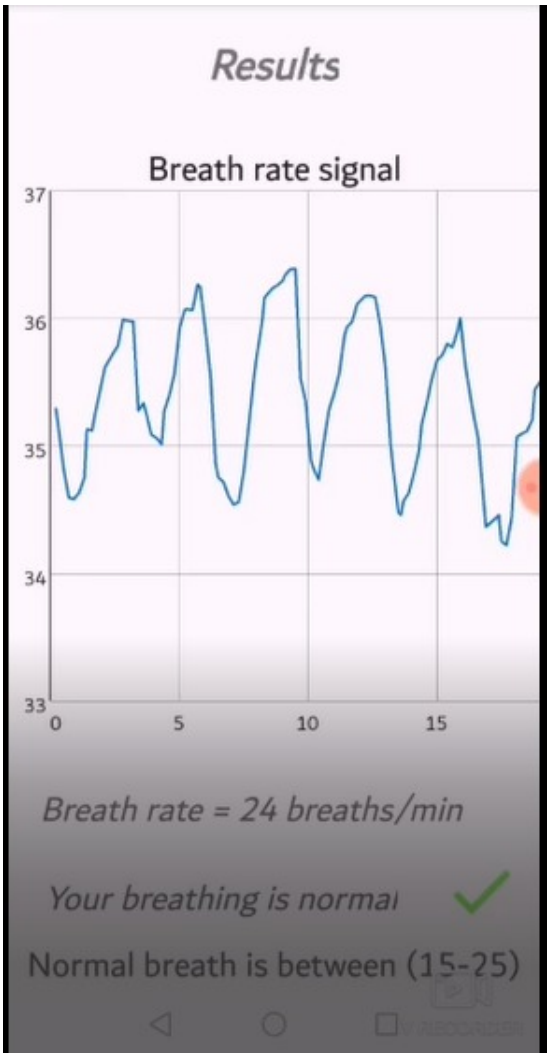

FIGURE 2.11: Result page API for normal breath rate

Figure 2.12 is for a user with abnormal breath rate and you can clearly see that it displays text that the calculated breath rate is abnormal and shows the alert sign.

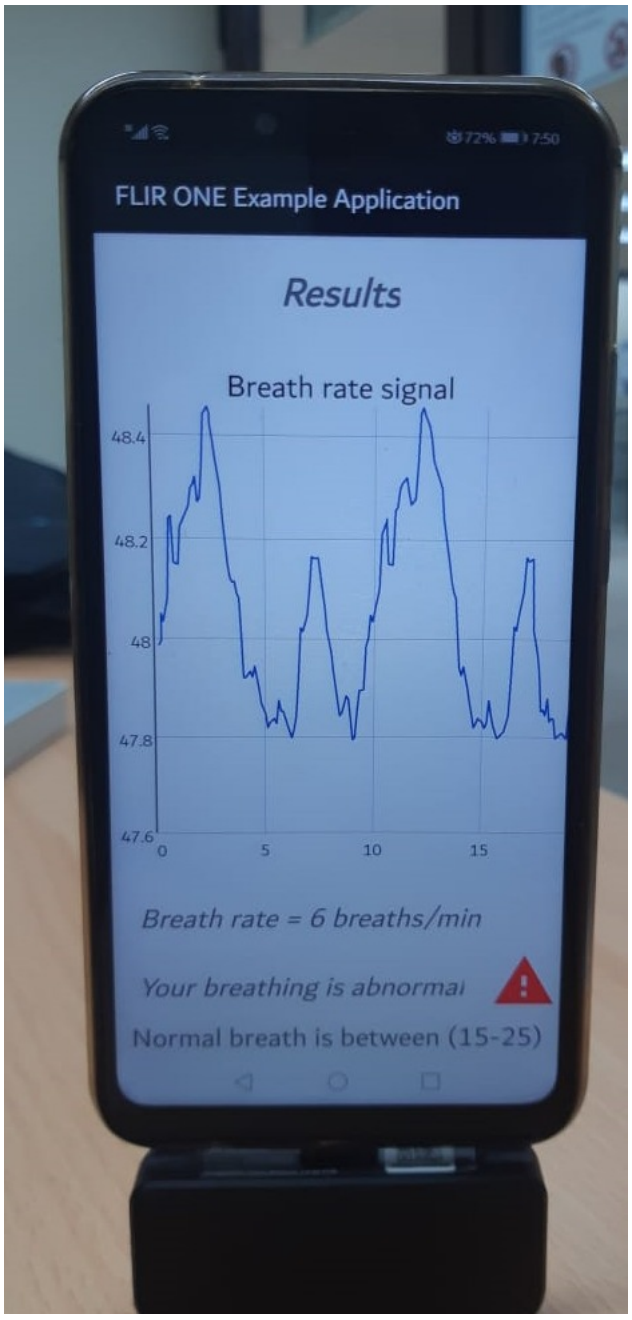

FIGURE 2.12: Result page API for abnormal breath rate

Chapter 3

Project Management

3.1 Team Members

Our team consists of two computer engineering students and an Electrical engineering student. Mohammed Ezat, Ahmed Sanad Ahmed and Haitham Chabayta. We introduce below the team members in details and highlight their strengths, weaknesses and their areas of contribution.

3.1.1 Mohammed Ezzat

Mohammed is our team leader. He is responsible for assigning weekly tasks for the team members and coordinating the team. Mohammed has an excellent background in back-end development, database management, and artificial intelligence.
However, his weakness is that he is not very patient and isn't satisfied until all the tasks are done. Mohammed did also attend every meeting we had with our supervisor once every two weeks and he organized the meeting we had between the team members only once a week, while also assigning the upcoming task for each student during the meeting. This member's main task was to test out different open CV libraries for image processing and automatic facial recognition.

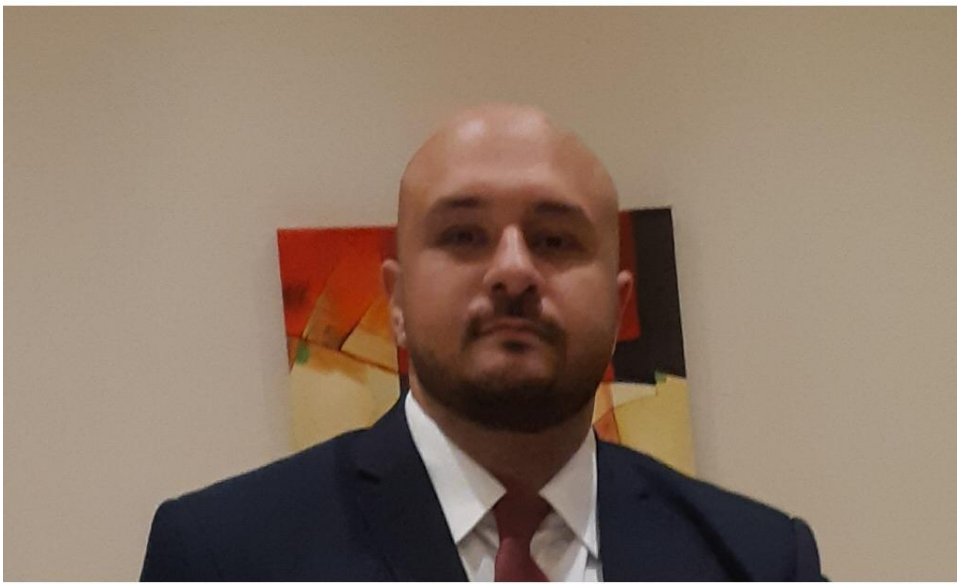

FIGURE 3.1: Mohammed Ezzat

3.1.2 Ahmed Sanad Ahmed

Ahmed is an electrical engineering student with a strong background in circuit design and analysis. He is also experienced with many electrical instruments and engineering softwares, the likes of Matlab and Multisim. In addition, Ahmed has great speaking and writing skills because he loves poetry. His only weakness though is that he doesn't have a great deal of knowledge in many programming languages. Ahmed is always available and excited to work. He did attend almost every meeting we had with our supervisor and in the weekly team meeting he always offered new ideas to enhance our reports and presentations. His main responsibility was to study the algorithm we will use to measure the respiratory rate and detect sleep apnea from the recorded breathing temperatures.

FIGURE 3.2: Ahmed Sanad Ahmed

3.1.3 Haitham Chabayta

Haitham has an excellent background in web development as he is experienced in markup languages, java script, and bootstrap. He also developed a good knowledge in building API's. Haitham has also had a professional work experience for over a year which enhanced his communication skills. His weakness though is that he is not too experienced with electrical equipment. He is always available for work and he did attend every team meeting we had with our supervisor. As for the team meetings we had every week between the team members only to analyze our progress and assign new tasks to each member, he always contributes positively to those meetings through presenting new ideas and solutions. This member's main task was to process the video obtained from the mobile thermal camera, frame by frame, plot the breath rate signal and to test the camera's SDK on android studio.

FIGURE 3.3: Haitham Chabayta

3.2 Gantt Chart

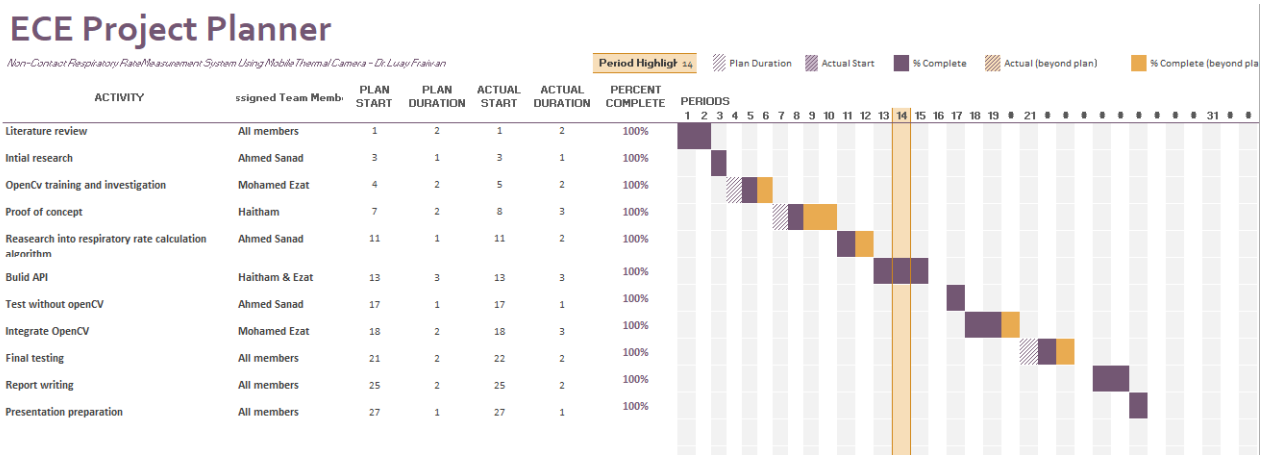

FIGURE 3.4: Gant Chart

Here are the objectives and goals that we intend to meet throughout our project:

- To detect the breathing pattern of a subjects in an indoor environment using a mobile thermal camera.
- To measure The Respiratory Rate accurately
- To detect the breathing abnormalities and mainly infant sleep Apnea.
- To provide a real time continuous monitoring of infant breathing rate and temperature.
- To design a convenient and user friendly mobile application to measure respiratory rate.
- Provide a low cost solution for measuring respiratory rate.

3.3 Tasks

Since Mohammed Ezat was the leader, he assigned the tasks in our group. Our project divided into three phases and in all phases each member has a specific task to do it. Phase one aim is to get a clear idea about the concept of the project and to explore some research that relate on the project topic. Ahmed Sanad worked on the initial research and discus that with the group members. Mohammed and Haitham explore about open CV and android studio. In phase two, Ahmed Sanad test the respiratory rate algorithm while Haitham researched on automatic facial recognition and build an API. Mohammed worked on the implementation of the automatic facial recognition.

Finally, in phase three we had finalize the application and this was done by all group members at the same time.

3.4 Team Communication Methods

During the capstone course, we had face some difficulties that slow down our progress in the project, but we were able to communicate between us through several methods that will be shown in this section.

3.4.1 Meetings

During the capstone course, we were able to arrange meetings with our supervisor Dr. Luay Fraiwan by booking an appointment through the email. Dr. Luay was friendly , helpful and he give us the opportunity to earn some skills from his long experience in this field. A few sample of our discussion is shown in Fig.3.5 Fig.3.6 Fig.3.7

3.4.2 WhatsApp

We had made a WhatsApp group to share our ideas and discuss the project progress. The WhatsApp group was very useful because it is the easiest method to communicate between the group members. Fig.3.8 shows a sample of our discussion in WhatsApp.

3.4.3 Microsoft Teams

Due to COVID-19 risks, the Ministry of Education had decided to close the universities in the country. That was very difficult challenge for us to continue our progress in the project and to communicate between us and the supervisor. Microsoft Teams make the things simpler and we had done several meetings with the supervisor Dr. Luay Fraiwan through Microsoft Teams. In Fig.3.9 a sample of our meetings.

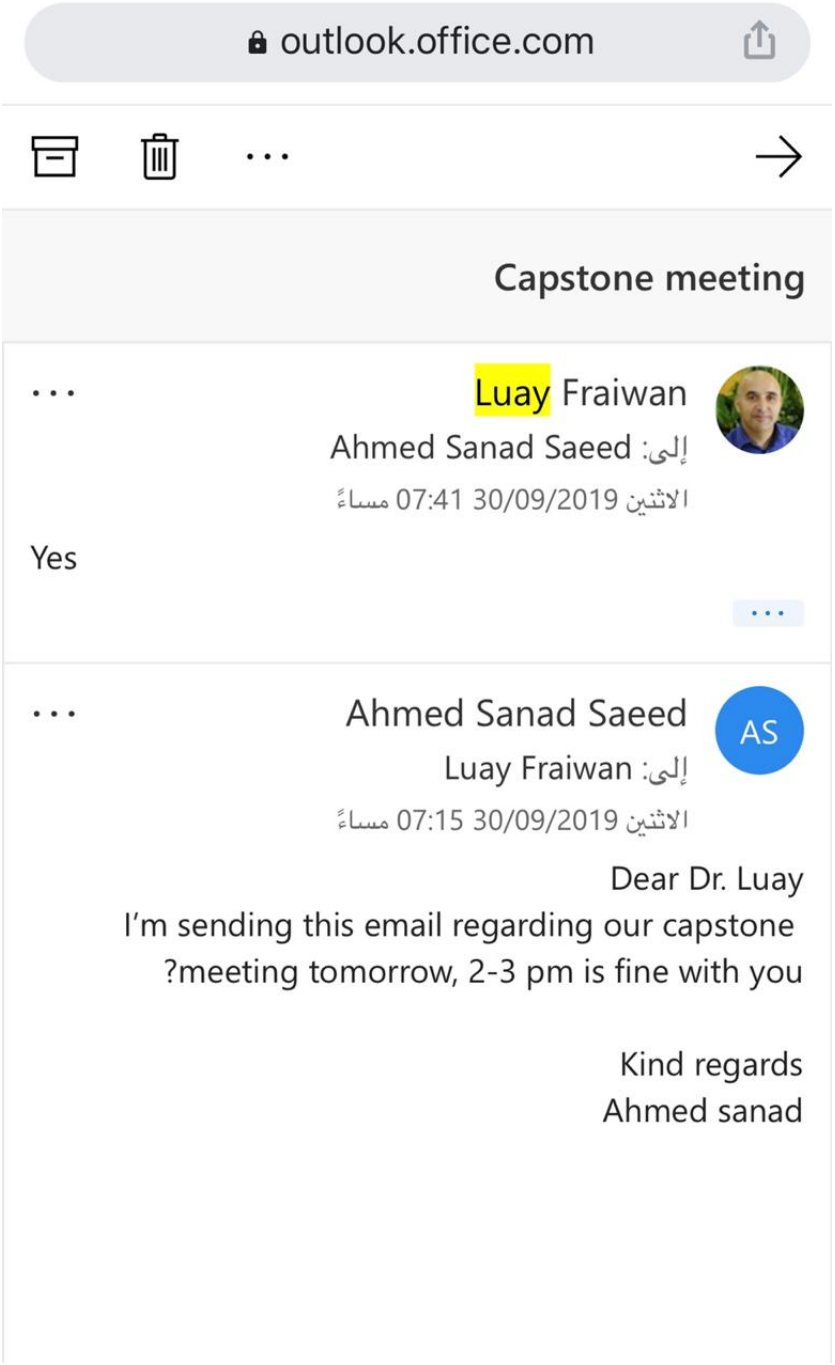

FIGURE 3.5: Meetings sample 1

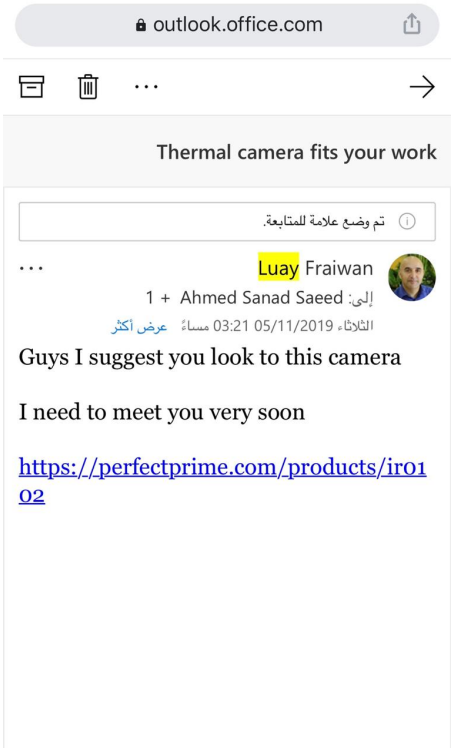

FIGURE 3.6: Meetings sample 2

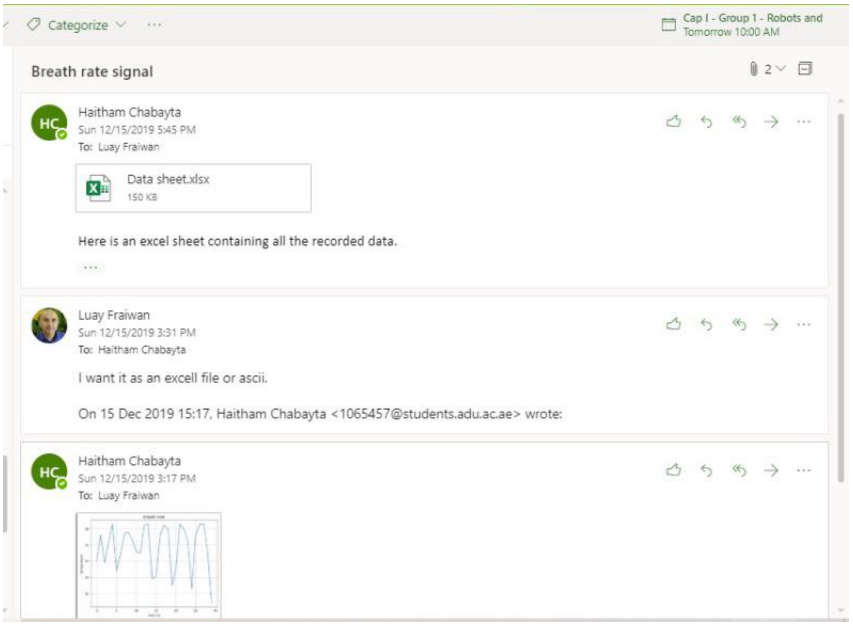

FIGURE 3.7: Meetings sample 3

FIGURE 3.8: Whatsapp communication sample

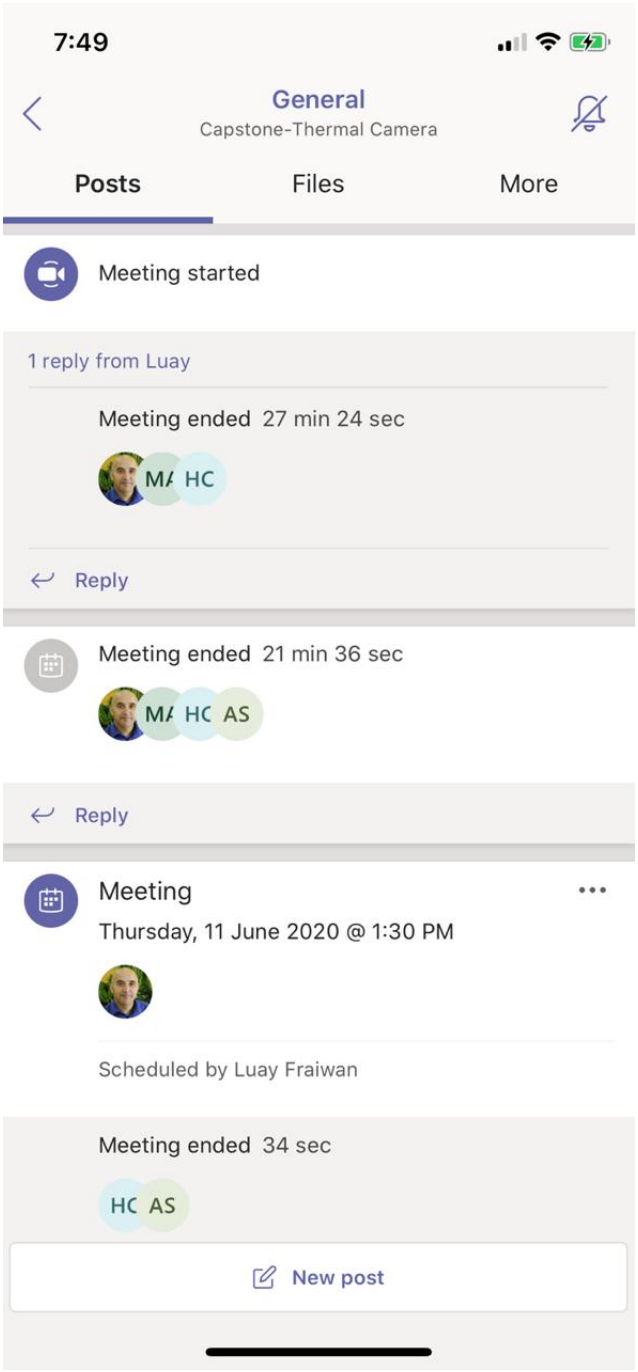

FIGURE 3.9: Teams meeting sample

Chapter 4

Results and Discussion

As we have discussed before there are 3 approaches that we have considered in developing the application and our testing plan involves conducting first a simple test in idle conditions for each approach and then to preform a test on the different distances and angles each approach can operate accurately from.

4.0.1 First approach results

For first approach we monitor the average temperature of an area, continuously for 30 seconds, that is defined manually and not detected automatically. The expected result of our system is a smooth sine wave like signal, and as we can see from the figure 4.1 below the first approach does succeed in providing result that are as expected. We can clearly see from the signal that it has smooth edges and behaves similarly to a sine wave.

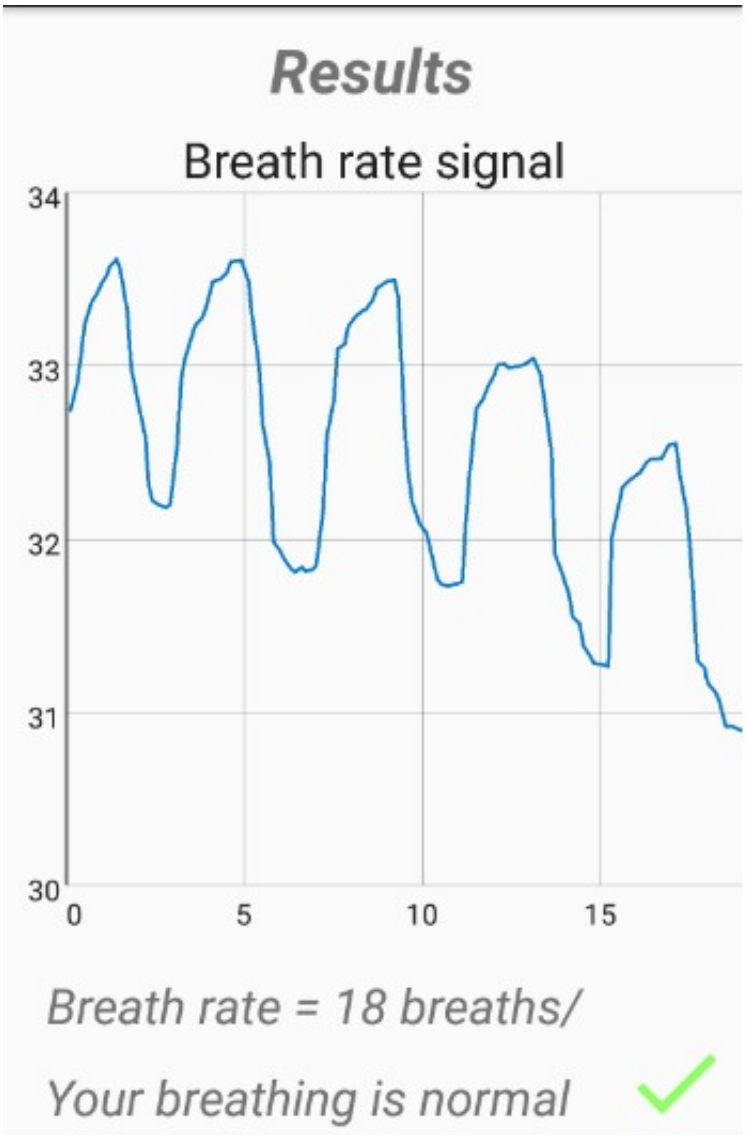

FIGURE 4.1: Approach 1 Idle test result

We also expect to obtain a normal respiratory rate as we are conducting the test on an adult in rest so the expected respiratory rate calculation outcome should be in between 15 and 25 breaths per min. The calculated respiratory rate should also be inline with the number of peaks present in the respiration signal and as shown in fig 4.1 we have 6 peaks in the respiration signal in 20 seconds, therefore the breath rate should be 6 * 3 which is 18, and the breath rate we calculated was also equal to 18. The results of the first approach are promising and inline with what is expected.

4.0.2 Second approach results

For the second approach, we will also monitor the average temperature of an area for 30 seconds, but this area is detected automatically and it is the area under the nose of the user. We will detect this area continuously throughout the 30 seconds period. The expected result stays the same; a smooth sine wave like signal. figure 4.2 shows the resulted respiration signal for the second approach. As clearly shown the result signal is not as expected as it has very sharp edges.

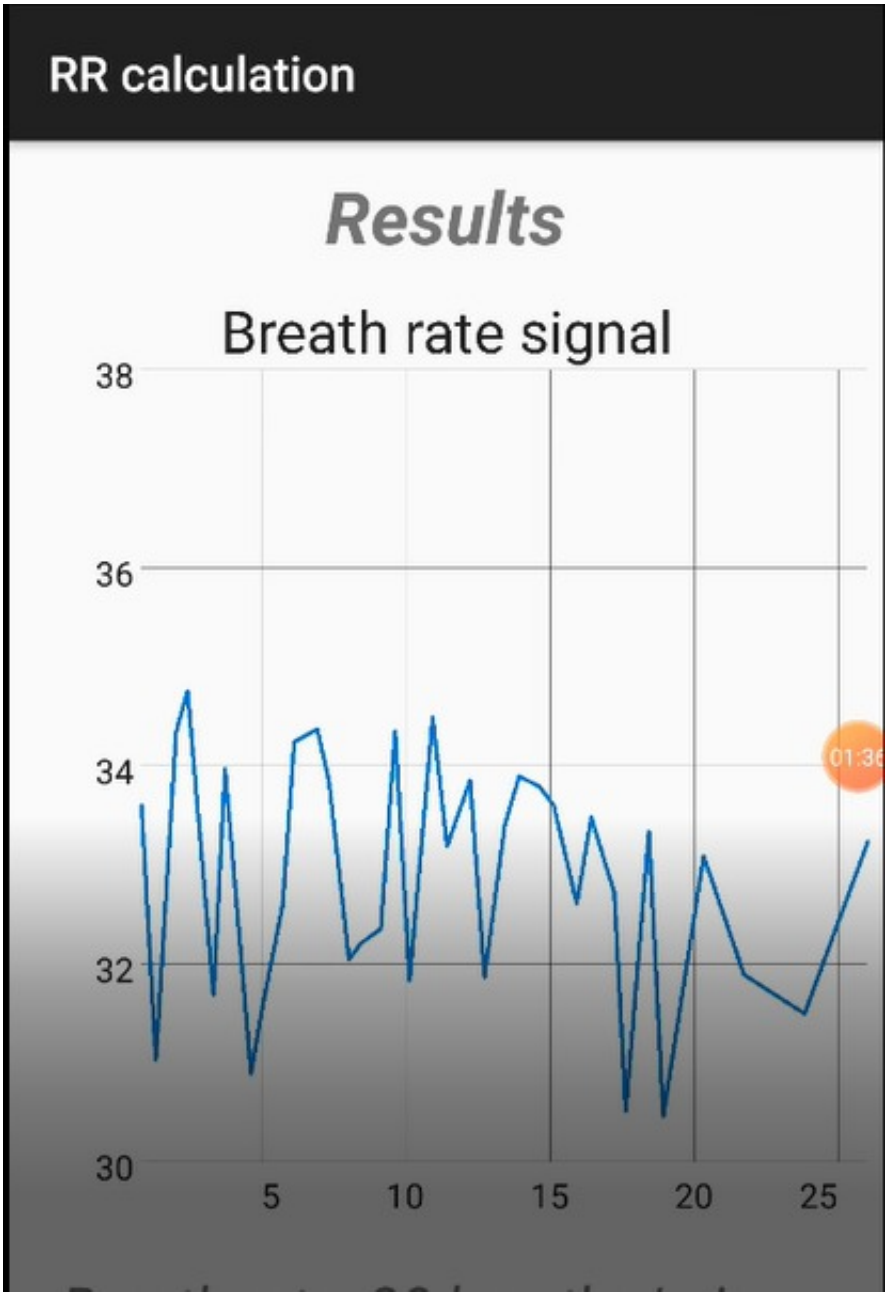

FIGURE 4.2: Approach 2 Idle test result

That is due to the openCV detection takes around half a second to execute and so while the openCV is processing many frames are lost and the average number of frames we are capturing went down to 2 frames per second from the normal 9 frames per second the Flir one mobile thermal camera provides. Due to the inaccuracies in the signal resulted the calculate respiratory rate will also be inaccurate and therefore the second approach result were not inline with what is expected and doesn't meet the acceptance criteria of the test conducted.

4.0.3 Third approach results

The third approach is similar to the second approach, as we will measure the average temperature of an area that is detected automatically using openCV for 30 seconds. However, the difference is in approach 3 we will not detect the area under the nose continuously instead, we will detect it only at regular intervals of 5 seconds. Figure 4.3 below shows the respiration signal resulted using the third approach 3.

FIGURE 4.3: Approach 3 Idle test result

As we the other two approaches the signal is still expected to be smooth and similar to a sine wave, and displayed in fig 4.3, the signal obtained is smooth and behaves like a sine wave. That is because in the third approach we will only perform openCV processing 6 times during the 30 seconds period of recording as we are only detecting the area under the nose each 5 seconds. Therefore only few number of frames are lost and the respiration signal obtained is accurate.

4.0.4 Distance and angle testing results

It is important to identify the acceptable distance for the user to be away from the phone were the application can still function correctly and it is also important to define the angle deviation range, from where the user is facing the phone, that will not affect the results. Several tests for the distance and angle are conducted on the first and third approach only, as the second approach did not provide acceptable result in idle conditions. After several tests conducted to measure the distance range were the application can operate for both approaches, We have found that approach 3 operates at a more flexible distance range (15-50 cm) while for approach 1 it is 15-30 cm. Approach 1 operates at a more limited range that is because the area defined, manually, to measure its average temperature is static and if the user is far the from phone the static area defined will cover more than the user's area under his or her nose, while for approach 3 it detects the exact area under the nose automatically and can do it for a user up to 50 cm away from the phone.

For the angle deviation acceptable for the camera from the user's face, several test were also conducted to determine that angle deviation range acceptable. The test results showed that the angle deviation is also more flexible for third approach in compassion to the first approach. Approach 3 can provide accurate results for users that are facing the camera from any angle between about 0 to 25 or -25 degrees angle, while for the first approach it is between only 0 to 15 or -15 degrees. That is also due to the area being dynamic and detected automatically in the third approach while it is static in the first approach. figure 4.4 below shows how the resulted signal looks like for a test conducted to a user, between 30 - 50 cm away from the camera or from an angle between 15 -25 degrees, using the first approach, while figure 4.5 shows the resulted signal for the same condition but using the third approach. We can clearly see that the signal resulted using approach is a valid respiration signal, while the respiration signal obtained from approach 1, figure 4.4, is invalid and provides false reading for the breathing temperatures.

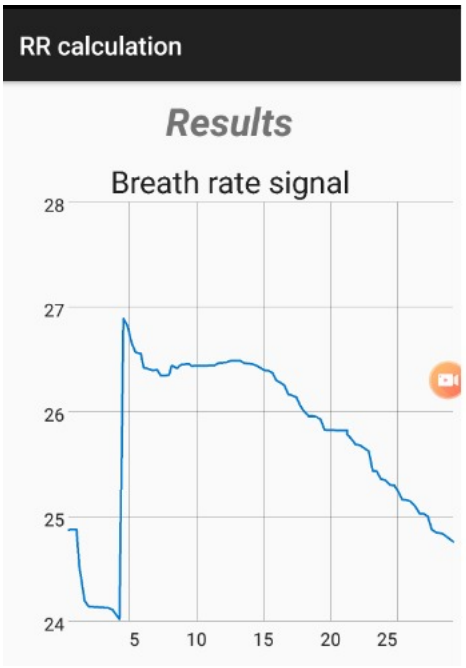

FIGURE 4.4: Respiration signal for approach 1 (Distance: 30-50cm)

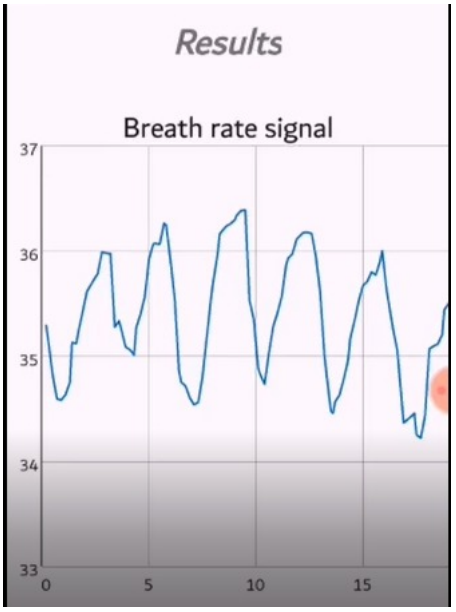

FIGURE 4.5: Respiration signal for approach 3 (Distance: 30-50cm)

Due to the test results we have to decide to develop the final version of our application based on the third approach, and because it is more convenient for the user for the application to automatically detect the area where we need to measure the average temperature.

Chapter 5

Conclusion and Future Work

5.1 Summary

The diagnosis of breathing disorder such as sleep apnea cost a lot and with the ever-expending life responsibilities and technology, we knew that we can come up with a solution that saves time and money for such a dangerous disorder. People who are diagnosed with such a disorder or who knows any loved one diagnosed with it, know how important it is to simplify the diagnosis process that is currently used in the real world. We aimed to come up with a simple and innovative solution that could be used by anybody. Therefore, we created a user-friendly mobile application that could detect breathing disorders on any platform by anybody with just a few clicks on their phones. Throughout our project, we encountered many challenges, especially while working on the mobile thermal camera SDK and learning how to process thermal images through Android studio and open CV, which was a first to us. Not only we discovered and learned new techniques, but we also further developed skills we already had in different programming languages, including Java, and engineering software's through constant use and practice. Moreover, our project contributes heavily to the era of smart solutions and technology as it simplifies user life through the power of technology.

5.2 Future work

Improvement can be done to our project by using a better and more expensive mobile thermal camera that provides more resolution and frames per second, as the camera we used in our projects provides us only with 480 x 640 resolution and 9 frames per second. Another thing that could be done to enhance our system is to develop a web page and link it to the application to provide live feed back to doctors on the patients conditions

on the web page. One more improvement could be done is to offer a save option on the results page in the application for user's to keep track of their test results records.

5.3 Lessons learned

5.3.1 Technical lessons

In this project, we learned some new skills in mobile application development. we also learned how to use the FLIR one mobile thermal camera and FLIR one SDK. Also, we become more familiar with open CV face recognition libraries which helped us enhance our project and made it more convenient for the user. Moreover, we understood how to design a user-friendly interface and how to analysis and diagnose signals through Java and Android SDK.

5.3.2 Teamwork Lessons

During the different stages of our project, we learned how to communicate effectively between each other and how to divide the tasks between us in a more time-efficient manner. Furthermore, we also experienced how working as one team helps us obtain fast results and a better outcome. We also learned how to tolerate the differences between us as we are a group coming from different majors, and how to integrate the different knowledge we gained to implement our project in the best possible way. Also, Due to the COVID-19 outbreak, we learned how to communicate effectively online and how to share work, ideas, and hold constructive meetings without the need to meet physically.

Appendix A

Non contact respiratory rate measurement application code

Main activity java code

```
package com.flir.flironeexampleapplication;

import android.Manifest;
import android.annotation.TargetApi;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.content.res.Resources;
import android.graphics.Bitmap;
import android.graphics.Color;
import android.graphics.ColorFilter;
import android.opengl.GLSurfaceView;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v4.content.PermissionChecker;
import android.util.AttributeSet;
import android.util.Log;
import android.view.MotionEvent;
import android.view.OrientationEventListener;
import android.view.ScaleGestureDetector;
import android.view.SurfaceView;
import android.view.View;
import android.widget.AbsoluteLayout;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.LinearLayout;
```

```
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ToggleButton;

import com.flir.flironeexampleapplication.util.SystemUiHider;
import com.flir.flironesdk.Device;
import com.flir.flironesdk.FlirUsbDevice;
import com.flir.flironesdk.Frame;
import com.flir.flironesdk.FrameProcessor;
import com.flir.flironesdk.RenderedImage;
import com.flir.flironesdk.SimulatedDevice;
import com.jjoe64.graphview.series.DataPoint;

import org.opencv.android.BaseLoaderCallback;
import org.opencv.android.CameraBridgeViewBase;
import org.opencv.android.JavaCameraView;
import org.opencv.android.LoaderCallbackInterface;
import org.opencv.android.OpenCVLoader;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.core.MatOfByte;
import org.opencv.core.MatOfRect;
import org.opencv.core.Point;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;
import org.opencv.objdetect.CascadeClassifier;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.Socket;
import java.nio.ByteBuffer;
import java.text.NumberFormat;
import java.util.ArrayList;
import java.util.EnumSet;
import java.util.LinkedList;
import java.util.Map;
import java.util.Timer;
import java.util.TimerTask;

/**
 * An example activity and delegate for FLIR One image streaming and device interaction.
 * Based on an example full-screen activity that shows and hides the system UI (i.e.
 * status bar and navigation/system bar) with user interaction.
 *
 * @see SystemUiHider
 * @see Device.Delegate
 * @see FrameProcessor.Delegate
 * @see Device.StreamDelegate
 * @see Device.PowerUpdateDelegate
 */
```

```
public class GLPreviewActivity extends Activity implements Device.Delegate, FrameProcessor.Deleg

    LinearLayout linearLayout;
    CascadeClassifier faceDetector;
    private Mat mrgba , mGrey;
    File cascadeFile;


    GLSurfaceView thermalSurfaceView;
    private volatile boolean startRecording = false;
    double fractionOfSecond = 0 ;
    int seconds = 0;
    private volatile Socket streamSocket = null;
    private boolean chargeCableIsConnected = true;
    ArrayList<DataPoint> avrFrameTemp = new ArrayList<>();

    private int deviceRotation= 0;
    private OrientationEventListener orientationEventListener;


    private volatile Device flirOneDevice;
    private FrameProcessor frameProcessor;


    private Device.TuningState currentTuningState = Device.TuningState.Unknown;
    private boolean accFrame = true;


    double rectX = 200;
    double rectY = 400;
    double rectHight = 1;
    double rectWidth = 1;

    double rectXpxl = 200;
    double rectYpxl = 400;
    double rectHightpxl = 1;
    double rectWidthpxl = 1;
    ArrayList<Frame> oldFrames = new ArrayList<Frame>();
    ArrayList<Double> oldFramesTime = new ArrayList<>();

    boolean addFrame = true;
    boolean calculated = false;
    boolean stopAcc = false;
    boolean accessOpenCV = true;
    // Device Delegate methods

    // Called during device discovery, when a device is connected
    // During this callback, you should save a reference to device
    // You should also set the power update delegate for the device if you have one
    // Go ahead and start frame stream as soon as connected, in this use case
    // Finally we create a frame processor for rendering frames

    public void onDeviceConnected(Device device){
        Log.i("ExampleApp", "Device connected!");
        runOnUiThread(new Runnable() {
```

```
            @Override
            public void run() {
                findViewById(R.id.pleaseConnect).setVisibility(View.GONE);
            }
        });

        flirOneDevice = device;
        flirOneDevice.setPowerUpdateDelegate(this);
        flirOneDevice.startFrameStream(this);

        final ToggleButton chargeCableButton = (ToggleButton)findViewById(R.id.chargeCableToggle
        if(flirOneDevice instanceof SimulatedDevice){
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    chargeCableButton.setChecked(chargeCableIsConnected);
                    chargeCableButton.setVisibility(View.VISIBLE);
                }
            });
        }else{
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    chargeCableButton.setChecked(chargeCableIsConnected);
                    chargeCableButton.setVisibility(View.INVISIBLE);
                    findViewById(R.id.connect_sim_button).setEnabled(false);

                }
            });
        }

        orientationEventListener.enable();
    }

    /**
     * Indicate to the user that the device has disconnected
     */
    public void onDeviceDisconnected(Device device){
        Log.i("ExampleApp", "Device disconnected!");

        final ToggleButton chargeCableButton = (ToggleButton)findViewById(R.id.chargeCableToggle
        final TextView levelTextView = (TextView)findViewById(R.id.batteryLevelTextView);
        final ImageView chargingIndicator = (ImageView)findViewById(R.id.batteryChargeIndicator)
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                findViewById(R.id.pleaseConnect).setVisibility(View.GONE);
                levelTextView.setText("--");
                chargeCableButton.setChecked(chargeCableIsConnected);
                chargeCableButton.setVisibility(View.INVISIBLE);
                chargingIndicator.setVisibility(View.GONE);
                findViewById(R.id.tuningProgressBar).setVisibility(View.GONE);
                findViewById(R.id.tuningTextView).setVisibility(View.GONE);
                findViewById(R.id.connect_sim_button).setEnabled(true);
            }
```

```
        });
        flirOneDevice = null;
        orientationEventListener.disable();
    }

    /**
     * If using RenderedImage.ImageType.ThermalRadiometricKelvinImage, you should not rely on
     * the accuracy if tuningState is not Device.TuningState.Tuned
     * @param tuningState
     */
    public void onTuningStateChanged(Device.TuningState tuningState){

        currentTuningState = tuningState;
        if (tuningState == Device.TuningState.InProgress){
            runOnUiThread(new Thread(){
                @Override
                public void run() {
                    super.run();
                    findViewById(R.id.tuningProgressBar).setVisibility(View.VISIBLE);
                    findViewById(R.id.tuningTextView).setVisibility(View.VISIBLE);
                }
            });
        }else {
            runOnUiThread(new Thread() {
                @Override
                public void run() {
                    super.run();
                    findViewById(R.id.tuningProgressBar).setVisibility(View.GONE);
                    findViewById(R.id.tuningTextView).setVisibility(View.GONE);
                }
            });
        }
    }

    @Override
    public void onAutomaticTuningChanged(boolean deviceWillTuneAutomatically) {

    }
    private ColorFilter originalChargingIndicatorColor = null;
    @Override
    public void onBatteryChargingStateReceived(final Device.BatteryChargingState batteryCharging

        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                ImageView chargingIndicator = (ImageView)findViewById(R.id.batteryChargeIndicato
                if (originalChargingIndicatorColor == null){
                    originalChargingIndicatorColor = chargingIndicator.getColorFilter();
                }
                switch (batteryChargingState) {
                    case FAULT:
                    case FAULT_HEAT:
                        chargingIndicator.setColorFilter(Color.RED);
                        chargingIndicator.setVisibility(View.VISIBLE);
                        break;
```

```
                case FAULT_BAD_CHARGER:
                    chargingIndicator.setColorFilter(Color.DKGRAY);
                    chargingIndicator.setVisibility(View.VISIBLE);
                case MANAGED_CHARGING:
                    chargingIndicator.setColorFilter(originalChargingIndicatorColor);
                    chargingIndicator.setVisibility(View.VISIBLE);
                    break;
                case NO_CHARGING:
                default:
                    chargingIndicator.setVisibility(View.GONE);
                    break;
            }
        }
    });
}
@Override
public void onBatteryPercentageReceived(final byte percentage){
    Log.i("ExampleApp", "Battery percentage received!");

    final TextView levelTextView = (TextView)findViewById(R.id.batteryLevelTextView);
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            levelTextView.setText(String.valueOf((int) percentage) + "%");
        }
    });


}

// StreamDelegate method
public void onFrameReceived(Frame frame) {
    Log.v("ExampleApp", "Frame received!");
    // Frame nframe  = null;




    if (currentTuningState != Device.TuningState.InProgress){
       /*
        if(this.startRecording){
            this.oldFrames.add(frame);
            double time = fractionOfSecond/10.0;
            this.oldFramesTime.add(time);
        }
        */


        if (accFrame ) {
            frameProcessor.processFrame(frame, FrameProcessor.QueuingOption.CLEAR_QUEUED);
            thermalSurfaceView.requestRender();
        }
```

```
    }



}

private Bitmap thermalBitmap = null;

// Frame Processor Delegate method, will be called each time a rendered frame is produced
public void onFrameProcessed(final RenderedImage renderedImage){
    try {

        if(renderedImage.imageType() == RenderedImage.ImageType.VisibleAlignedRGBA8888Image
            if(startRecording)
                stopAcc= true;
            accFrame = false;
            mrgba = new Mat(renderedImage.width() , renderedImage.height(), CvType.CV_8UC3);
            //mrgba =  Imgcodecs.imdecode(new MatOfByte(renderedImage.pixelData()), Imgcodec
            mrgba.put(0,0, renderedImage.pixelData());
            final MatOfRect faceDetection = new MatOfRect();
            faceDetector.detectMultiScale(mrgba , faceDetection);

            double width = renderedImage.width();
            double height = renderedImage.height();
            double screenX = Resources.getSystem().getDisplayMetrics().widthPixels;
            double screenY = Resources.getSystem().getDisplayMetrics().heightPixels;

            double opencvWidth = mrgba.width();
            double opencvHeight = mrgba.height();


            for (Rect rect:faceDetection.toArray()){

                rectX = (rect.x +(rect.x * (1/5.0)))  * (screenX/opencvWidth) ;
                rectXpxl = rect.y +(rect.x * (2/5.0));
                rectY = (rect.y + (rect.height/2.0)) * (screenY/opencvHeight) ;
                rectYpxl = rect.x + (rect.width/2.0);
                rectHight = (rect.height /2.0) * (screenX/opencvWidth);
                rectHightpxl = rect.width*0.6;
                rectWidth = (rect.width * 0.8) *(screenX/opencvWidth);
                rectWidthpxl = rect.height* 0.8 ;
            }


            /*
            TextView T1 = ((TextView)findViewById(R.id.spotMeterValue));
            String bla = opencvWidth + " " + opencvHeight ;
            T1.setText(bla);*/

            AbsoluteLayout rect = (AbsoluteLayout)findViewById(R.id.recDetect);
            rect.setX((int)rectX);
```

```
            rect.setY((int)rectY);
            rect.getLayoutParams().height = (int)rectHight;
            rect.getLayoutParams().width = (int)rectWidth;
           /*
            if(faceDetection.toArray().length == 0){
                rect.setX(0);
                rect.setY(0);
                rect.getLayoutParams().height = 0;
                rect.getLayoutParams().width = 0;

                 rectXpxl = 0 ;
                 rectYpxl = 0;
                 rectHightpxl = 0;
                 rectWidthpxl = 0;
            }*/


            accFrame = true;
        }
    }catch (final Exception ex ){
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                ((TextView)findViewById(R.id.spotMeterValue)).setText("error "+ex.getMessage
            }
        });
    }


    if (renderedImage.imageType() == RenderedImage.ImageType.ThermalRadiometricKelvinImage &
        // Note: this code is not optimized

        int[] thermalPixels = renderedImage.thermalPixelValues();
        // average the center 9 pixels for the spot meter

        int theWidth = renderedImage.width();

        int startPixelIndex = (int)(((rectYpxl-1) * (rectWidthpxl)) + rectXpxl);
        int indexexNo = (int)(rectWidthpxl*rectHightpxl);
        int[] centerPixelIndexes = new int[indexexNo];
        int newWidth = (int) rectWidthpxl;

        for(int x = 0 ; x < centerPixelIndexes.length ; x++){
            centerPixelIndexes[x] = startPixelIndex +x;

            if (x %newWidth == 0 && x !=0){
                startPixelIndex += (int)(theWidth - rectWidthpxl);
            }

        }

        double averageTemp = 0;

        for (int i = 0; i < centerPixelIndexes.length; i++){
            // Remember: all primitives are signed, we want the unsigned value,
```

```
                // we've used renderedImage.thermalPixelValues() to get unsigned values

                int pixelValue = (thermalPixels[centerPixelIndexes[i]]);

                averageTemp += (((double)pixelValue) - averageTemp) / ((double) i + 1);
            }
            double averageC = (averageTemp / 100) - 273.15;
            double second = fractionOfSecond/10.0;
            DataPoint tempPoint = new DataPoint(second , averageC);
            avrFrameTemp.add(tempPoint);

            NumberFormat numberFormat = NumberFormat.getInstance();
            numberFormat.setMaximumFractionDigits(2);
            numberFormat.setMinimumFractionDigits(2);

            String spotMeterValue = numberFormat.format(averageC) + " C ";
            TextView textView = ((TextView)findViewById(R.id.spotMeterValue));
            textView.setText(spotMeterValue);
            stopAcc= false;
        }




}

/**
 * Whether or not the system UI should be auto-hidden after
 * {@link #AUTO_HIDE_DELAY_MILLIS} milliseconds.
 */
private static final boolean AUTO_HIDE = true;

/**
 * If {@link #AUTO_HIDE} is set, the number of milliseconds to wait after
 * user interaction before hiding the system UI.
 */
private static final int AUTO_HIDE_DELAY_MILLIS = 3000;

/**
 * If set, will toggle the system UI visibility upon interaction. Otherwise,
 * will show the system UI visibility upon interaction.
 */
private static final boolean TOGGLE_ON_CLICK = true;

/**
 * The flags to pass to {@link SystemUiHider#getInstance}.
 */
private static final int HIDER_FLAGS = SystemUiHider.FLAG_HIDE_NAVIGATION;

/**
 * The instance of the {@link SystemUiHider} for this activity.
 */
private SystemUiHider mSystemUiHider;
public void onTuneClicked(View v){
```

```
        if (flirOneDevice != null){
            flirOneDevice.performTuning();
        }

    }

    public void onCaptureImageClicked(View v){
        accessOpenCV = false;
        if(flirOneDevice != null) {
            this.startRecording = true;
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    ((ImageButton)findViewById(R.id.imageButton)).setEnabled(false);
                }
            });

            new Timer().scheduleAtFixedRate(new TimerTask(){
                @Override
                public void run(){
                    fractionOfSecond++;
                    if(fractionOfSecond %10 ==0){
                        seconds++;

                        if(seconds %5 == 0)
                            accessOpenCV = true;
                        else
                            accessOpenCV = false;

                        if(seconds<10) {
                            runOnUiThread(new Runnable() {

                                @Override
                                public void run() {
                                    ((TextView) findViewById(R.id.SecondsPreview)).setText("0:0"
                                }
                            });
                        }
                        else {
                            runOnUiThread(new Runnable() {

                                @Override
                                public void run() {
                                    ((TextView) findViewById(R.id.SecondsPreview)).setText("0:"
                                }
                            });
                        }

                    }

                    if(seconds == 30){
                        //addFrame = false;
                        runOnUiThread(new Runnable() {
                            @Override
                            public void run() {
```

```
                        ((ImageButton)findViewById(R.id.imageButton)).setEnabled(true);
                    }
                });


                seconds = 0 ;
                fractionOfSecond =0;
                startRecording = false;
                calculated = true;

                try {
                    /*
                    if(accFrame){
                        Frame nframe = oldFrames.remove(0);
                        cTime = oldFramesTime.remove(0);
                        frameProcessor.processFrame(nframe, FrameProcessor.QueuingOption
                        thermalSurfaceView.requestRender();
                    }*/
                    calculateBreathRate();
                    this.cancel();
                    /*
                    if(oldFrames.size() == 0){

                    }*/

                }catch (final Exception ex){
                    runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            ((TextView)findViewById(R.id.spotMeterValue)).setText("error
                        }
                    });
                }

            }
        }
    },0,100);
    }
}


public void onConnectSimClicked(View v){
    if(flirOneDevice == null){
        try {
            flirOneDevice = new SimulatedDevice(this, this, getResources().openRawResource(R
            flirOneDevice.setPowerUpdateDelegate(this);
            chargeCableIsConnected = true;
        } catch(Exception ex) {
            flirOneDevice = null;
            Log.w("FLIROneExampleApp", "IO EXCEPTION");
            ex.printStackTrace();
        }
    }else if(flirOneDevice instanceof SimulatedDevice) {
        flirOneDevice.close();
        flirOneDevice = null;
```

```
        }
    }

    public void onSimulatedChargeCableToggleClicked(View v){
        if(flirOneDevice instanceof SimulatedDevice){
            chargeCableIsConnected = !chargeCableIsConnected;
            ((SimulatedDevice)flirOneDevice).setChargeCableState(chargeCableIsConnected);
        }
    }

    public void onRotateClicked(View v){
        ToggleButton theSwitch = (ToggleButton)v;
        if (theSwitch.isChecked()){
            frameProcessor.setImageRotation(180.0f);
        }else{
            frameProcessor.setImageRotation(0.0f);
        }
    }

    public void onChangeViewClicked(View v){
        if (frameProcessor == null){
            ((ToggleButton)v).setChecked(false);
            return;
        }
        ListView paletteListView = (ListView)findViewById(R.id.paletteListView);
        ListView imageTypeListView = (ListView)findViewById(R.id.imageTypeListView);
        if (((ToggleButton)v).isChecked()){
            // only show palette list if selected image type is colorized
            paletteListView.setVisibility(View.INVISIBLE);
            for (RenderedImage.ImageType imageType : frameProcessor.getImageTypes()){
                if (imageType.isColorized()) {
                    paletteListView.setVisibility(View.VISIBLE);
                    break;
                }
            }
            imageTypeListView.setVisibility(View.VISIBLE);
            findViewById(R.id.imageTypeListContainer).setVisibility(View.VISIBLE);
        }else{
            findViewById(R.id.imageTypeListContainer).setVisibility(View.GONE);
        }


    }

    public void onImageTypeListViewClicked(View v){
        int index = ((ListView) v).getSelectedItemPosition();
        RenderedImage.ImageType imageType = RenderedImage.ImageType.values()[index];
        frameProcessor.setGLOutputMode(imageType);
        int paletteVisibility = (imageType.isColorized()) ? View.VISIBLE : View.GONE;
        findViewById(R.id.paletteListView).setVisibility(paletteVisibility);
    }

    public void onPaletteListViewClicked(View v){
        RenderedImage.Palette pal = (RenderedImage.Palette )(((ListView)v).getSelectedItem());
        frameProcessor.setImagePalette(pal);
```

```
}

/**
 * Example method of starting/stopping a frame stream to a host
 * @param v The toggle button pushed
 */
public void onVividClicked(View v){
    final ToggleButton button = (ToggleButton)v;
    frameProcessor.setVividIrEnabled(button.isChecked());
}

@Override
protected void onStart(){
    super.onStart();
    mGrey = new Mat();
    mrgba = new Mat();
    if (Device.getSupportedDeviceClasses(this).contains(FlirUsbDevice.class)){
        findViewById(R.id.pleaseConnect).setVisibility(View.VISIBLE);
    }
    try {
        Device.startDiscovery(this, this);
    }catch(IllegalStateException e){
        // it's okay if we've already started discovery
    }catch (SecurityException e){
        // On some platforms, we need the user to select the app to give us permisison to th
        Toast.makeText(this, "Please insert FLIR One and select "+getString(R.string.app_nam
        // There is likely a cleaner way to recover, but for now, exit the activity and
        // wait for user to follow the instructions;
        finish();
    }
}

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

    setContentView(R.layout.activity_gl_preview);


    final View controlsView = findViewById(R.id.fullscreen_content_controls);
    final View controlsViewTop = findViewById(R.id.fullscreen_content_controls_top);
    final View contentView = findViewById(R.id.fullscreen_content);
    linearLayout = findViewById(R.id.fullscreen_content_controls_top);


    RenderedImage.ImageType defaultImageType = RenderedImage.ImageType.BlendedMSXRGBA8888Ima
    frameProcessor = new FrameProcessor(this, this, EnumSet.of(RenderedImage.ImageType.Therm
    frameProcessor.setGLOutputMode(defaultImageType);


    thermalSurfaceView = (GLSurfaceView) findViewById(R.id.imageView);
    thermalSurfaceView.setPreserveEGLContextOnPause(true);
    thermalSurfaceView.setEGLContextClientVersion(2);
    thermalSurfaceView.setRenderer(frameProcessor);
```

```
thermalSurfaceView.setRenderMode(GLSurfaceView.RENDERMODE_WHEN_DIRTY);
thermalSurfaceView.setDebugFlags(GLSurfaceView.DEBUG_CHECK_GL_ERROR | GLSurfaceView.DEBU


final String[] imageTypeNames = new String[]{ "Visible", "Thermal", "MSX" };
final RenderedImage.ImageType[] imageTypeValues = new RenderedImage.ImageType[]{
        RenderedImage.ImageType.VisibleAlignedRGBA8888Image,
        RenderedImage.ImageType.ThermalRGBA8888Image,
        RenderedImage.ImageType.BlendedMSXRGBA8888Image,
};

ListView imageTypeListView = ((ListView)findViewById(R.id.imageTypeListView));
imageTypeListView.setAdapter(new ArrayAdapter<>(this,R.layout.emptytextview,imageTypeNam
imageTypeListView.setSelection(defaultImageType.ordinal());
imageTypeListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        if (frameProcessor != null) {
            RenderedImage.ImageType imageType = imageTypeValues[position];
            frameProcessor.setGLOutputMode(imageType);
            if (imageType.isColorized()){
                findViewById(R.id.paletteListView).setVisibility(View.VISIBLE);
            }else{
                findViewById(R.id.paletteListView).setVisibility(View.INVISIBLE);
            }
        }
    }
});
imageTypeListView.setDivider(null);

// Palette List View Setup
ListView paletteListView = ((ListView)findViewById(R.id.paletteListView));
paletteListView.setDivider(null);
paletteListView.setAdapter(new ArrayAdapter<>(this, R.layout.emptytextview, RenderedImag
paletteListView.setSelection(frameProcessor.getImagePalette().ordinal());
paletteListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        if (frameProcessor != null){
            frameProcessor.setImagePalette(RenderedImage.Palette.values()[position]);
        }
    }
});
// Set up an instance of SystemUiHider to control the system UI for
// this activity.

mSystemUiHider = SystemUiHider.getInstance(this, contentView, HIDER_FLAGS);
mSystemUiHider.setup();
mSystemUiHider
        .setOnVisibilityChangeListener(new SystemUiHider.OnVisibilityChangeListener() {
            // Cached values.
            int mControlsHeight;
            int mShortAnimTime;

            @Override
```

```
            @TargetApi(Build.VERSION_CODES.HONEYCOMB_MR2)
            public void onVisibilityChange(boolean visible) {
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB_MR2) {
                    // If the ViewPropertyAnimator API is available
                    // (Honeycomb MR2 and later), use it to animate the
                    // in-layout UI controls at the bottom of the
                    // screen.
                    if (mControlsHeight == 0) {
                        mControlsHeight = controlsView.getHeight();
                    }
                    if (mShortAnimTime == 0) {
                        mShortAnimTime = getResources().getInteger(
                                android.R.integer.config_shortAnimTime);
                    }
                    controlsView.animate()
                            .translationY(visible ? 0 : mControlsHeight)
                            .setDuration(mShortAnimTime);
                    controlsViewTop.animate().translationY(visible ? 0 : -1 * mControlsH
                } else {
                    // If the ViewPropertyAnimator APIs aren't
                    // available, simply show or hide the in-layout UI
                    // controls.
                    controlsView.setVisibility(visible ? View.VISIBLE : View.GONE);
                    controlsViewTop.setVisibility(visible ? View.VISIBLE : View.GONE);
                }

                if (visible && !((ToggleButton)findViewById(R.id.change_view_button)).is
                    // Schedule a hide().
                    delayedHide(AUTO_HIDE_DELAY_MILLIS);
                }
            }
        });

        // Set up the user interaction to manually show or hide the system UI.
        contentView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (TOGGLE_ON_CLICK) {
                    mSystemUiHider.toggle();
                } else {
                    mSystemUiHider.show();
                }
            }
        });

        // Upon interacting with UI controls, delay any scheduled hide()
        // operations to prevent the jarring behavior of controls going away
        // while interacting with the UI.
        findViewById(R.id.change_view_button).setOnTouchListener(mDelayHideTouchListener);


        orientationEventListener = new OrientationEventListener(this) {
            @Override
            public void onOrientationChanged(int orientation) {
                deviceRotation = orientation;
```

```
        }
    };
    mScaleDetector = new ScaleGestureDetector(this, new ScaleGestureDetector.OnScaleGestureL
        @Override
        public void onScaleEnd(ScaleGestureDetector detector) {
        }
        @Override
        public boolean onScaleBegin(ScaleGestureDetector detector) {
            return true;
        }
        @Override
        public boolean onScale(ScaleGestureDetector detector) {
            Log.d("ZOOM", "zoom ongoing, scale: " + detector.getScaleFactor());
            frameProcessor.setMSXDistance(detector.getScaleFactor());
            return false;
        }
    });

    findViewById(R.id.fullscreen_content).setOnTouchListener(new View.OnTouchListener() {
        @Override
        public boolean onTouch(View v, MotionEvent event) {
            mScaleDetector.onTouchEvent(event);
            return true;
        }
    });

    // ask for permission?
    String writePermission = Manifest.permission.WRITE_EXTERNAL_STORAGE;
    boolean permissionGranted = false;
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        permissionGranted = (ContextCompat.checkSelfPermission(this, writePermission) == Pac
    }
    else {
        permissionGranted = (PermissionChecker.checkSelfPermission(this, writePermission) ==
    }

    if(!permissionGranted) {
        if (ActivityCompat.shouldShowRequestPermissionRationale(this, writePermission)) {
            Toast.makeText(this, "App requires write permission to save photos", Toast.LENGT
        }
        else {
            ActivityCompat.requestPermissions(this, new String[]{writePermission}, 0);
        }
    }

    if(!OpenCVLoader.initDebug()){
        OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_4_0, this,baseCallback);

    }else{
        try {
            baseCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);

        } catch (IOException e) {

            runOnUiThread(new Runnable() {
```

```
                @Override
                public void run() {
                    ((TextView)findViewById(R.id.spotMeterValue)).setText("manager dose not
                }
            });

        }

    }


}

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

    thermalSurfaceView.onPause();
    if (flirOneDevice != null){
        flirOneDevice.stopFrameStream();
    }
}
@Override
public void onResume(){
    super.onResume();

    thermalSurfaceView.onResume();

    if (flirOneDevice != null){
        flirOneDevice.startFrameStream(this);
    }
}
@Override
public void onStop() {
    // We must unregister our usb receiver, otherwise we will steal events from other apps
    Log.e("PreviewActivity", "onStop, stopping discovery!");
    mrgba.release();
    mGrey.release();
    Device.stopDiscovery();
    flirOneDevice = null;
    super.onStop();
}

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

    // Trigger the initial hide() shortly after the activity has been
    // created, to briefly hint to the user that UI controls
    // are available.
    delayedHide(100);
}


/**
```

```
 * Touch listener to use for in-layout UI controls to delay hiding the
 * system UI. This is to prevent the jarring behavior of controls going away
 * while interacting with activity UI.
 */
View.OnTouchListener mDelayHideTouchListener = new View.OnTouchListener() {
    @Override
    public boolean onTouch(View view, MotionEvent motionEvent) {
        if (AUTO_HIDE) {
            delayedHide(AUTO_HIDE_DELAY_MILLIS);
        }
        return false;
    }
};

Handler mHideHandler = new Handler();
Runnable mHideRunnable = new Runnable() {
    @Override
    public void run() {
        mSystemUiHider.hide();
    }
};

/**
 * Schedules a call to hide() in [delay] milliseconds, canceling any
 * previously scheduled calls.
 */
private void delayedHide(int delayMillis) {
    mHideHandler.removeCallbacks(mHideRunnable);
    mHideHandler.postDelayed(mHideRunnable, delayMillis);
}


public void calculateBreathRate(){

    Intent intent = new Intent(GLPreviewActivity.this , resultsPreview.class);
    intent.putExtra("resultsList" , avrFrameTemp);
    startActivity(intent);

}

@Override
public void onCameraViewStarted(int width, int height) {
    /**
     mGrey = new Mat();
     mrgba = new Mat();**/
}

@Override
public void onCameraViewStopped() {
    /**
     mrgba.release();
     mGrey.release();**/
}

@Override
```

```
public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame) {
    /**mrgba = inputFrame.rgba();
     mGrey = inputFrame.gray();

     MatOfRect faceDetection = new MatOfRect();
     faceDetector.detectMultiScale(mrgba , faceDetection);

     for (Rect rect:faceDetection.toArray()){
     Imgproc.rectangle(mrgba ,
     new Point(rect.x , rect.y +(rect.height/2)),
     new Point(rect.x+rect.width, rect.y +rect.height),
     new Scalar(255,0,0));
     }
     return mrgba;**/
    return mrgba;
}


private BaseLoaderCallback baseCallback = new BaseLoaderCallback(this) {
    @Override
    public void onManagerConnected(int status) throws IOException {

        switch (status){
            case LoaderCallbackInterface.SUCCESS:{


                InputStream is = getResources().openRawResource(R.raw.haarcascade_mcs_nose);
                File cascadeDir = getDir("cascade" , Context.MODE_PRIVATE);
                cascadeFile = new File(cascadeDir , "haarcascade_mcs_nose.xml");

                FileOutputStream fos = new FileOutputStream(cascadeFile);
                byte[] buffer = new byte[4096];
                int bytesRead;

                while ((bytesRead = is.read(buffer)) != -1){
                    fos.write(buffer , 0 , bytesRead);
                }
                is.close();
                fos.close();
                //is.close();
                faceDetector = new CascadeClassifier((cascadeFile.getAbsolutePath()));
                if(faceDetector.empty()){
                    faceDetector=null;
                }else
                    cascadeDir.delete();
                // javaCameraView.enableView();
                //setContentView(thermalSurfaceView);
                //mrgba  = new Mat();


            }break;

            default:
            {
                super.onManagerConnected(status);
```

```
                }


            }

        }
    };
}
```

Main activity layout xml code

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
             xmlns:tools="http://schemas.android.com/tools"
             android:layout_width="match_parent"
             android:layout_height="match_parent"
             xmlns:app="http://schemas.android.com/apk/res-auto"
             android:background="@android:color/background_dark"
             android:orientation="vertical"
             tools:context=".GLPreviewActivity"
             android:keepScreenOn="true"
             android:id="@+id/topView"
             android:touchscreenBlocksFocus="false">




    <!-- This FrameLayout insets its children based on system windows using
         android:fitsSystemWindows. -->
    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="center">

        <android.opengl.GLSurfaceView
            android:id="@+id/imageView"
            android:layout_width="0dp"
            android:layout_height="0dp"
            android:layout_gravity="center_horizontal|top"
            android:focusable="false"
            android:focusableInTouchMode="false"
            android:scaleType="fitCenter"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintDimensionRatio="3:4"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.0"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <TextView
            android:id="@+id/textView3"
            android:layout_width="324px"
            android:layout_height="144px"
            android:background="@drawable/border"
            app:layout_constraintBottom_toBottomOf="@+id/imageView"
            app:layout_constraintEnd_toEndOf="parent"
```

```
            app:layout_constraintStart_toStartOf="@+id/imageView"
            app:layout_constraintTop_toTopOf="@+id/imageView"
            app:layout_constraintVertical_bias="0.499" />

        <TextView
            android:id="@+id/SecondsPreview"
            android:layout_width="94dp"
            android:layout_height="67dp"
            android:padding="10dp"
            android:textSize="36sp"
            android:textStyle="bold|italic"
            app:layout_constraintBottom_toBottomOf="@+id/imageView"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.499"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/textView3"
            app:layout_constraintVertical_bias="0.726" />

    </android.support.constraint.ConstraintLayout>


    <FrameLayout android:layout_width="match_parent" android:layout_height="match_parent"
        android:fitsSystemWindows="true"
        android:id="@+id/fullscreen_content">
        <LinearLayout
            android:id="@+id/fullscreen_content_controls_top" style="@android:style/ButtonBar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="top|center_horizontal" android:orientation="horizontal"
            tools:ignore="UselessParent"
            android:background="@color/black_overlay">
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:text="@string/battery_label"
                android:id="@+id/batteryLabelTextView"
                android:layout_margin="8dp"
                android:layout_weight="0"
                android:layout_gravity="left|center_vertical"
                style="?actionMenuTextAppearance"/>

            <ImageView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/batteryChargeIndicator"
                android:layout_gravity="left|center_vertical"
                android:layout_weight="0"
                android:src="@android:drawable/ic_lock_idle_charging"
                android:tint="@color/accent_material_light"
                android:visibility="gone" />

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
```

```
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:id="@+id/batteryLevelTextView"
            android:layout_weight="0.48"
            android:layout_gravity="left|center_vertical"
            android:text="--" />

        <ToggleButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Cable"
            android:id="@+id/chargeCableToggle"
            android:layout_weight="0.10"
            android:checked="false"
            android:visibility="invisible"
            android:onClick="onSimulatedChargeCableToggleClicked" />

        <ToggleButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/vividButton"
            style="?metaButtonBarButtonStyle"
            android:textOff="@string/normal"
            android:textOn="@string/vivid"
            android:checked="true"
            android:onClick="onVividClicked"/>

        <Button
            android:layout_width="51dp" style="?metaButtonBarButtonStyle"
            android:layout_height="wrap_content"
            android:text="@string/performTuning"
            android:id="@+id/tuneButton"
            android:onClick="onTuneClicked" />

        <ToggleButton
            android:layout_width="47dp"
            android:layout_height="wrap_content"
            android:text="@string/rotate_on"
            android:id="@+id/switch_rotate"
            android:onClick="onRotateClicked"
            android:layout_gravity="right"
            android:textOff="@string/rotate_off"
            android:textOn="@string/rotate_on"
            style="?metaButtonBarButtonStyle" />


    </LinearLayout>
    <FrameLayout android:id="@+id/fullscreen_content_controls" style="?metaButtonBarStyle"
        android:layout_width="match_parent" android:layout_height="wrap_content"
        android:layout_gravity="bottom|center_horizontal"
        android:background="@color/black_overlay"
        tools:ignore="UselessParent"
        android:columnCount="3"
        android:rowCount="2">
        <LinearLayout
            android:orientation="horizontal"
```

```
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal|bottom"
        android:baselineAligned="false"
        android:id="@+id/imageTypeListContainer"
        android:focusableInTouchMode="false"
        android:visibility="gone">

        <ListView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/imageTypeListView"
            android:layout_gravity="center_vertical|bottom|left"
            android:layout_marginBottom="60dp"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:alpha="0.9"
            android:layout_weight="0.25"
            android:choiceMode="singleChoice"
            android:clickable="true"
            android:dividerHeight="0dp" />

        <ListView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/paletteListView"
            android:layout_marginBottom="60dp"
            android:alpha="0.9"
            android:layout_gravity="bottom|right"
            android:layout_weight="0.75"
            android:choiceMode="singleChoice"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:clickable="true"
            android:dividerHeight="0dp" />
    </LinearLayout>

    <ImageButton
        android:id="@+id/imageButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal|bottom"
        android:layout_weight="1"
        android:src="@drawable/ic_fiber_manual_record_black_24dp"
        android:onClick="onCaptureImageClicked" />

    <ToggleButton android:id="@+id/change_view_button" style="?metaButtonBarButtonStyle"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_weight="0.33" android:textOff="@string/change_view"
        android:textOn="@string/change_view_retract"
        android:onClick="onChangeViewClicked"
        android:layout_gravity="bottom|left" />

    <Button
        android:layout_width="wrap_content" style="?metaButtonBarButtonStyle"
        android:layout_height="wrap_content"
        android:text="@string/connectSim"
```

```
            android:id="@+id/connect_sim_button"
            android:onClick="onConnectSimClicked"
            android:layout_gravity="center_horizontal|bottom|right" />


    </FrameLayout>

    <ProgressBar
        style="?android:attr/progressBarStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/tuningProgressBar"
        android:layout_gravity="center"
        android:visibility="gone" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="@string/txtTuning"
        android:id="@+id/tuningTextView"
        android:labelFor="@id/tuningProgressBar"
        android:layout_gravity="center"
        android:layout_marginTop="36dp"
        android:visibility="gone" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Please Connect FLIR One"
        android:id="@+id/pleaseConnect"
        android:layout_gravity="center"
        android:layout_marginTop="90dp"
        android:visibility="gone" />


</FrameLayout>


<TextView
    android:id="@+id/spotMeterValue"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:alpha="0.9"
    android:shadowColor="#000000"
    android:shadowDx="0"
    android:shadowDy="0"
    android:shadowRadius="3"
    android:textAppearance="?android:attr/textAppearanceSmall" />


<AbsoluteLayout
    android:id="@+id/recDetect"
```

```
        android:layout_width="0px"
        android:layout_height="0px"
        android:background="@drawable/rectangle"
        ></AbsoluteLayout>



</FrameLayout>
```

Results activity java code

```
package com.flir.flironeexampleapplication;

import android.media.Image;
import android.support.annotation.DrawableRes;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ImageView;
import android.widget.TextView;

import com.jjoe64.graphview.GraphView;
import com.jjoe64.graphview.series.DataPoint;
import com.jjoe64.graphview.series.LineGraphSeries;

import java.util.ArrayList;

public class resultsPreview extends AppCompatActivity {
    private ArrayList<DataPoint> rsltArr ;
    LineGraphSeries<DataPoint> series;
    int peakCount = 0;

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

        rsltArr = (ArrayList<DataPoint>) getIntent().getSerializableExtra("resultsList");

        double x, y ;
        x =0 ;

        GraphView graph = (GraphView)findViewById(R.id.signalView);
        ImageView tickView = (ImageView)findViewById(R.id.tickView);

        series = new LineGraphSeries<DataPoint>();

        series.setTitle("Breath rate");

        double nextNextTemp, nextTemp, currentTemp;
        ArrayList path = new ArrayList();
        String prevState = "";
        String currentState = "" ;


        final double lastTime = rsltArr.get(rsltArr.size()-1).getX();
```

```
for (int i =0 ; i< rsltArr.size()-2; i++){
    series.appendData(rsltArr.get(i), false, 270);
    currentTemp = rsltArr.get(i).getY();
    nextTemp = rsltArr.get(i+1).getY();
    nextNextTemp = rsltArr.get(i+2).getY();
    if(currentTemp < nextTemp && nextTemp >nextNextTemp){
        peakCount++;
    }

}

graph.addSeries(series);

graph.setTitle("Breath rate signal");
graph.setTitleTextSize(72);

graph.setPadding(5, 5, 5, 5);
graph.getViewport().setScalable(true);

runOnUiThread(new Runnable() {
    @Override
    public void run() {
        ((TextView)findViewById(R.id.breathRateView)).setText("Breath rate = " + (peakCo
    }
});

if((peakCount*2)>25||(peakCount*2)<15) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            ((TextView) findViewById(R.id.resultReview)).setText("Your breathing is abno
        }
    });

    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            ((ImageView) findViewById(R.id.tickView)).setImageResource(R.drawable.ic_war
        }
    });

}

else {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            ((TextView) findViewById(R.id.resultReview)).setText("Your breathing is norm
        }
    });

    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            ((ImageView) findViewById(R.id.tickView)).setImageResource(R.drawable.ic_che
```

```
                }
            });

        }


    }
}
```

Results activity layout xml code

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".resultsPreview">

    <com.jjoe64.graphview.GraphView
        android:id="@+id/signalView"
        android:layout_width="wrap_content"
        android:layout_height="380dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.352" />

    <TextView
        android:id="@+id/breathRateView"
        android:layout_width="300dp"
        android:layout_height="38dp"
        android:padding="5dp"
        android:text="Breath rate: 18 breaths/min"
        android:textSize="24sp"
        android:textStyle="italic"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.27"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/signalView"
        app:layout_constraintVertical_bias="0.191" />

    <TextView
        android:id="@+id/header"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:text="Results"
        android:textSize="30sp"
        android:textStyle="bold|italic"
        app:layout_constraintBottom_toTopOf="@+id/signalView"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
```

```
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.253" />

    <TextView
        android:id="@+id/resultReview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingLeft="2dp"
        android:paddingTop="10dp"
        android:paddingRight="20dp"
        android:paddingBottom="10dp"
        android:text="Your breathing is normal"
        android:textSize="24sp"
        android:textStyle="italic"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.225"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/breathRateView"
        app:layout_constraintVertical_bias="0.287" />

    <TextView
        android:id="@+id/resultReview2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Normal breath is between (15-25)"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/resultReview"
        app:layout_constraintVertical_bias="0.302" />

    <ImageView
        android:id="@+id/tickView"
        android:layout_width="56dp"
        android:layout_height="53dp"
        android:src="@drawable/ic_check_black_24dp"
        app:layout_constraintBottom_toTopOf="@+id/resultReview2"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.428"
        app:layout_constraintStart_toEndOf="@+id/resultReview"
        app:layout_constraintTop_toBottomOf="@+id/breathRateView"
        app:layout_constraintVertical_bias="0.729" />


</android.support.constraint.ConstraintLayout>
```

Bibliography

[1] Naresh M Punjabi. The epidemiology of adult obstructive sleep apnea. *Proceedings of the American Thoracic Society*, 5(2):136–143, 2008.

[2] Anne G Wheaton, Geraldine S Perry, Daniel P Chapman, and Janet B Croft. Sleep disordered breathing and depression among us adults: National health and nutrition examination survey, 2005-2008. *Sleep*, 35(4):461–467, 2012.

[3] David TR Berry, Wilse B Webb, and A Jay Block. Sleep apnea syndrome: a critical review of the apnea index as a diagnostic criterion. *Chest*, 86(4):529–531, 1984.

[4] Heather M Engleman, James P McDonald, David Graham, Glenn E Lello, Ruth N Kingshott, Emma L Coleman, Thomas W Mackay, and Neil J Douglas. Randomized crossover trial of two treatments for sleep apnea/hypopnea syndrome: continuous positive airway pressure and mandibular repositioning splint. *American journal of respiratory and critical care medicine*, 166(6):855–859, 2002.

[5] Dr Farah Al-khalidi, Reza Saatchi, Derek Burke, Heather Elphick, and Stephen Tan. Respiration rate monitoring methods: A review. *Pediatric Pulmonology*, 46: 523 – 529, 06 2011. doi: 10.1002/ppul.21416.

[6] Sidney Ray. *Scientific photography and applied imaging.* Routledge, 1999.

[7] Richard Szeliski. *Computer vision: algorithms and applications.* Springer Science & Business Media, 2010.

[8] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008.

[9] Ashish Pant, Arjun Arora, Suneet Kumar, and RP Arora. Sophisticated image encryption using opencv. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(1), 2012.

[10] Robert Laganière. *OpenCV Computer Vision Application Programming Cookbook Second Edition.* Packt Publishing Ltd, 2014.

[11] Shervin Emami and Valentin Petrut Suciu. Facial recognition using opencv. *Journal of Mobile, Embedded and Distributed Systems*, 4(1):38–43, 2012.

[12] Android Studio. Meet android studio, June 2020. URL https://developer.android.com/studio/intro.

[13] M. Rouse. What is android studio, October 2018. URL https://searchmobilecomputing.techtarget.com/definition/Android-Studio.

[14] Belen Cruz Zapata. Android studio 2 essentials: a fast-paced guide to get you up and running with android application development using android studio 2. 2016.

[15] Farah Al-Khalidi, Reza Saatchi, Heather Elphick, and Derek Burke. An evaluation of thermal imaging based respiration rate monitoring in children. *American Journal of Engineering and Applied Sciences*, 4(4):586–597, 2011.

[16] Jacek Ruminski and Alicja Kwasniewska. Evaluation of respiration rate using thermal imaging in mobile conditions. In *Application of Infrared to Biomedical Sciences*, pages 311–346. Springer, 2017.

[17] Abdulkadir Hamidu Alkali, Reza Saatchi, Heather Elphick, and Derek Burke. Facial tracking in thermal images for real-time noncontact respiration rate monitoring. In *2013 European Modelling Symposium*, pages 265–270. Ieee, 2013.

[18] Barbara Gefvert. Thermal imaging solves respiration monitoring woes. 2014.

[19] Mario Carrillo. Solving real-world problems with mobile thermal imaging. 2019.

[20] Iley B Browning, Gilbert E D'Alonzo, and Martin J Tobin. Importance of respiratory rate as an indicator of respiratory dysfunction in patients with cystic fibrosis. *Chest*, 97(6):1317–1321, 1990.

[21] George Yuan, Nicole A Drost, and R Andrew McIvor. Respiratory rate and breathing pattern. *McMaster University Medical Journal*, 10(1):23–25, 2013.

[22] Ross C Kory. Routine measurement of respiratory rate: an expensive tribute to tradition. *Journal of the American Medical Association*, 165(5):448–450, 1957.

[23] Gilles Pagès and Jacques Printems. Optimal quadratic quantization for numerics: the gaussian case. *Monte Carlo methods and applications*, 9(2):135–165, 2003.

[24] Youngjun Cho, Simon J Julier, Nicolai Marquardt, and Nadia Bianchi-Berthouze. Robust tracking of respiratory rate in high-dynamic range scenes using mobile thermal imaging. *Biomedical optics express*, 8(10):4480–4503, 2017.

[25] Carina Barbosa Pereira, Xinchi Yu, Michael Czaplik, Rolf Rossaint, Vladimir Blazek, and Steffen Leonhardt. Remote monitoring of breathing dynamics using infrared thermography. *Biomedical optics express*, 6(11):4378–4394, 2015.

[26] Farah Q AL-Khalidi, Reza Saatchi, Derek Burke, H Elphick, and Stephen Tan. Respiration rate monitoring methods: A review. *Pediatric pulmonology*, 46(6): 523–529, 2011.

[27] Advantage environment. Thermal cameras save energy, reduce environmental impact. 2016.

[28] Nilanjan Dey, Amira S Ashour, and Afnan S Althoupety. Thermal imaging in medical science. In *Recent Advances in Applied Thermal Imaging for Industrial Applications*, pages 87–117. IGI Global, 2017.

[29] IK Lee, Chih-Chi Wang, Meng-Chih Lin, Chia-Te Kung, Kuo-Chung Lan, and Chien-Te Lee. Effective strategies to prevent coronavirus disease-2019 (covid-19) outbreak in hospital. *The Journal of Hospital Infection*, 105(1):102, 2020.

[30] 2020. ODILUPO, E. Tracking respiratory rate and the coronavirus. URL https://www.whoop.com/thelocker/respiratory-rate-tracking-coronavirus/.

[31] Consumer.huawei.com. n.d. huawei nova 3i — android phone. URL https://consumer.huawei.com/in/phones/nova3i.

[32] J. R. Hughes J. M. Cowan, J. M. Burris, end-expired breath temperature M. P. Cunningham, "The relationship of normal body temperature, and vol. 34 no. 5 pp. 238–242 2010. bac/brac ratio in 98 physically fit human test subjects, " Journal of analytical toxicology.

[33] Raj Rakshit A. K. O. T. C. D. G. a. P. Arijit Sinharay. "the ultrasonic directional tidal breathing pattern sensor: Equitable design realization based on phase information," tcs research and innovation, kolkata 700156, india, vol. 1853, pp. 1 – 21, 08 2017.

[34] Folkestad L B. J. B. M. Nielsen LG. "inter-observer agreement in measuring respiratory rate," plos one, p. 1, 06 2015.

[35] Developer.flir.com. 2018. Android platform guide – flir one/cat s60 – flir one developer. [online] available at: https://developer.flir.com/getting-started/android-platform-guide-flir-one-cat-s60/.

[36] Opencv.org. n.d. Android. 2020. URL https://opencv.org/android/.