

DRIVER DROWSINESS DETECTION SYSTEM

SUPERVISED BY

DR. HOSAM EL-OCLA

DEPARTMENT OF COMPUTER SCIENCE

LAKEHEAD UNIVERSITY



Submitted By

Vasista Avinash Behara

DEPARTMENT OF COMPUTER SCIENCE

Lakehead University,

955 Oliver Road, Thunder Bay, Ontario, Canada

September 2019

Acknowledgement

I take this opportunity to express my gratitude to my supervisor Dr Hosam El-Ocla, department of Computer Science, Lakehead University for his guidance and support during the project. I thank him for the technical insights that helped me in developing some of the features of the project. I appreciate his patience and critical criticism that helped me in pushing myself to learn the technology and complete the project.

I want to express my special gratitude and thanks to the Computer Science Graduate department at Lakehead University for giving me the wonderful opportunity to prove myself.

I would also like to thank my family for their constant support during the period of study. They not only supported me during a good time but also lifted and motivated me through difficult times.

ABSTRACT

With the advancement of science innovation and the vehicle industry, there is an ever-increasing number of vehicles out and about. Tiredness and Fatigue of drivers are among the noteworthy reasons for street mishaps. Consistently, they increment the measures of deaths and fatalities wounds all-inclusive. In this way, it is important to build up a strategy to identify the driver's sleepiness to decrease the mishap rates. To avoid this situation, this project proposes a sleepy eye recognition system for the detection of drowsiness. The algorithm used in this project determines the EAR (eye aspect ratio) using the Euclidean distances between the eyelids. This is done by using a facial landmark predictor which predicts facial landmarks. If the EAR is lower than a threshold value that is set in the algorithm for 5, 6 consecutive frames, then it is considered as a drowsy scenario. An alarm is triggered when the driver is drowsy, and an alert is sent to a phone number. This system is implemented in a raspberry pi for easy transit, and an android app is used to control the project. An android app is also used to control the raspberry pi. The communication between both the applications is done through a server. This project effectively monitors the driver's drowsiness and even works in bad lighting conditions.

Keywords : EAR, Raspberry pi, android, drowsiness, fatigue, detection, buzzer, SMS, blink, landmarks.

List of Contents

CHAPTER 1 INTRODUCTION	1
1.1 MOTIVATION AND BACKGROUND	1
1.2 LITERATURE REVIEW	2
CHAPTER 2 SYSTEM DESCRIPTION	4
2.1 RASPBERRY PI.....	4
2.1.1 SPECIFICATIONS	6
2.2 GENERAL PURPOSE I/O (GPIO)	7
2.2.1 Buzzer	8
2.2.2 Camera	8
2.3 SETUP	10
2.4 SOFTWARE REQUIREMENTS.....	11
2.5 HARDWARE REQUIREMENTS.....	11
2.6 ADVANTAGES	12
2.7 DISADVANTAGES	12
CHAPTER 3 PROJECT OVERVIEW	14
3.1 PROPOSED METHOD.....	14
3.2 FACIAL FEATURES AND LANDMARKS.....	14
3.3 EAR (EYE ASPECT RATIO)	18
3.4 DROWSINESS DETECTION.....	20
CHAPTER 4 ALGORITHM DESIGN AND ANALYSIS	22
4.1 FLOWCHART DIAGRAM.....	22
4.2 CODE.....	23
CHAPTER 5 IMPLEMENTATION	31
5.1 PROJECT SETUP	31
5.2 RUNNING THE SETUP	32
CHAPTER 6 RESULTS & PERFORMANCE ANALYSIS	40
6.1 EAR ANALYSIS	40
6.2 DIFFICULTIES AND LIMITATIONS	42
CHAPTER 7 CONCLUSION AND FUTURE WORK	43
6.1 CONCLUSION.....	43
6.2 FUTURE WORK	43
REFERENCES	45
APPENDIXES	47
APPENDIX A: MAIN.PY (RASPBERRY PI)	47
APPENDIX B: DROWSY.PY (RASPBERRY PI)	50
APPENDIX C: MAINACTIVITY.JAVA (ANDROID).....	53
APPENDIX D: ACTIVITY_MAIN.XML (ANDROID)	63
APPENDIX E: BASEACTIVITY.JAVA (ANDROID).....	66

List of Figures

Figure 2-1 Block Diagram of Raspberry pi [17]	5
Figure 2-2 Buzzer I/O	7
Figure 2-3 Buzzer [17]	8
Figure 2-4 USB Camera	9
Figure 2-5 IO representation of the raspberry pi setup	10
Figure 3-1 representation of 68 Facial Landmarks	15
Figure 3-2 Detecting facial Landmarks	16
Figure 3-3 eye landmarks	17
Figure 3-4 EAR Formula	18
Figure 3-5 EAR vs TIME	19
Figure 3-6 EAR detected when the app is launched	20
Figure 3-7 Drowsiness alert is shown when detected as drowsy	21
Figure 4-1 Flow Chart Diagram of the System	22
Figure 4-2 import packages [B]	23
Figure 4-3 Set IO Values for Raspberry pi [B]	24
Figure 4-4 EAR calculation [B]	24
Figure 4-5 Initialise values [B]	25
Figures 4-6 code to Calculate EAR [B]	25
Figure 4-7 Main algorithm [B] , [16]	26
Figure 4-8 request and post methods [A]	27
Figure 4-9 App design [D]	28
Figure 4-10 code to set power [A]	28
Figure 4-11 Access app values through files [A]	29
Figure 4-12 Set volume in the app [C]	29
Figure 4-13 Code to send SMS with a location in it [C]	30
Figure 5-1 Project working setup	31
Figure 5-2 Android Studio showing connected phone	32
Figure 5-3 App UI when everything is off [D]	33
Figure 5-4 Raspberry pi screen showing the program running [A]	34
Figure 5-5 App UI showing everything is on [D]	35
Figure 5-6 Raspberry pi screen showing updated information received from the app [A]	36
Figure 5-7 Raspberry pi screen showing video stream	37
Figure 5-8 Raspberry pi screen showing log details after detection	38
Figure 5-9 Screenshot showing SMS received	39
Figure 6-1 EAR vs Different people	41
Figures 6-2 3 images shown at different lighting conditions	42

List of Tables

Table 1 Software Requirements	11
Table 2 Hardware Requirements.....	11
Table 3 EAR of different people.....	40

Chapter 1

INTRODUCTION

1.1 Motivation and Background

Driver drowsiness is a significant factor in a large number of motor accidents. There are estimates that around 75000 road accidents that are fatigue-related annually [11]. These road accidents lead to deaths, fatal injuries, economic losses. Thus, it can be stated that a large number of accidents happen due to drowsiness and fatigue in drivers, and this needs to be addressed.

The development of technologies for detecting and preventing drowsiness is a major challenge in the field of accident avoidance. It is necessary to develop and improvise a system that can detect drowsiness of the drivers and implement it in the vehicles.

Accidents due to drowsy driving usually occurs in trucks, buses in the night. The usual reasons for the drowsiness in drivers are working in the night, having insomnia, taking medicines like cough syrups, people under the influence, etc. there are many ways to detect if the drivers are drowsy like head position, monitoring heart rate, eye detection, lane deviation, steering wheel detection, etc. For this project, I have implemented eye detection.

Most state-of-the-art systems use either active or passive ways to detect the eyes. Active systems often have expensive parts like an infrared camera, illuminators [2], wearable devices, glasses with close cameras, etc. Many ways and methods have been proposed to automatically detect drowsiness and blinking of eyes in a video sequence. Most of them

use a video detector that is made for this whole purpose. A little more insight into these will be given in the literature review. These types of setups impose strong requirements to detect the eyes. But nowadays, there are many robust real-time facial landmark detectors that capture most of the characteristics on the human face.

There are many ways to detect eyes, MATLAB is one of them. The processing capacities required by MATLAB were very high. Also, there were some problems with speed in real-time processing. MATLAB was capable of processing only 4-5 frames per second. This is even lower in a system with much lower ram like a raspberry pi.

Typically, where OpenCV came in. OpenCV is an open-source computer vision library. It is outlined for computational effectiveness and with a solid center on genuine-time applications. It makes a difference to construct advanced vision applications rapidly and effectively. OpenCV fulfilled the low processing control and high-speed prerequisites of our application. Dlib is a modern C++, python toolkit containing machine learning algorithms and tools for creating complex software in C++, python to solve real-world problems. It contains machine learning algorithms which are used to train the dataset to make predictions as reliable as possible.

1.2 Literature Review

[1] This project's main idea of using EAR to detect drowsiness is first introduced in this paper by Soukupova and Cech. They tried to make eye detection using landmarks and find an ear to show how much eye is opened or closed to detect blinks. Some people even tried to prevent computer vision syndromes [12],[13], and [14] . This project extends what they do to another extent and detect if the person is sleeping or not by finding their eyelids [15]. The expensive approaches that are mentioned before like [2] and [3] are not very viable as they use IR illuminator and close camera glasses that monitor eyes regularly respectively. These expensive implementations are voided by simply using a raspberry pi and a USB camera.

Different ways like Steering Pattern observation, Vehicle Position in Lane monitoring, Driver Eye/Face monitoring, and Physiological activity are instructed in paper [9] which will be used to detect the driver's sleepiness. This paper focuses on a model of smart Band that monitors the heart rate to detect the sleepiness. The grove ear clip that contains the device may be placed on the tip of the finger or the ear lobe. The device has a lightweight emitting diode and lightweight police investigation receivers like photodiode or light police investigation electrical device. This is a very hard implementation, but it is the setup is also hard to install and expensive.

Most of the progressive landmark detectors formulate a regression downside, wherever a mapping from a picture into landmark positions [6] or into different landmark parametrization [5] is learned. These fashionable landmark detectors are trained on “in-the-wild datasets” and that they are so sturdy to variable illumination, numerous facial expressions, and moderate non-frontal head rotations.

Moreover, they tried to make the existing state of the art systems fast, reliable (which was achieved), but they are still slow compared to the new implementations from [1] and [7]. An average error of the landmark localization of a progressive detector is typically below 5 PC of the inter-ocular distance. Recent ways run even considerably better in real-time [8]. Below is one example of the same scenario.

Facial segmentation model given in [7] is comparable to the projected technique. However, their system relies on active form models with a rumored interval of regarding five seconds per frame for the segmentation, and therefore the eye gap signal is normalized by statistics calculable by observant an extended sequence. The system is so useful for offline process solely. The system mentioned runs in real-time, so it is more reliable.

Chapter 2

SYSTEM DESCRIPTION

2.1 Raspberry Pi

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games. Here is a block diagram of a Raspberry pi in figure 2-1 which shows clear representation of where every part of Raspberry pi is located.

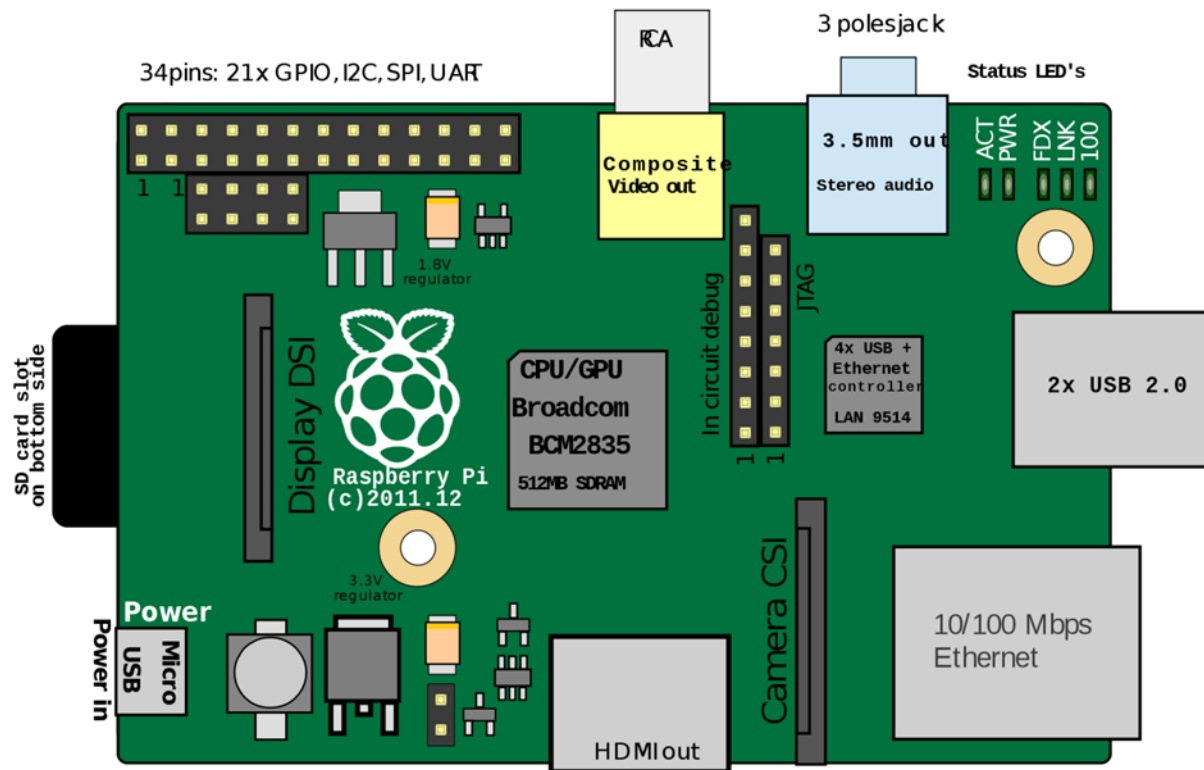


Figure 2-1 Block Diagram of Raspberry pi [17]

What's more, the Raspberry Pi can interact with the outside world and has been used in a wide array of digital maker projects, from music machines and parent detectors to weather stations and tweeting birdhouses with infra-red cameras. We want to see the Raspberry Pi being used by kids all over the world to learn to program and understand how computers work. There are currently four Raspberry Pi models. They are the Model A, the Model B, the Model B+, and the Compute Module. All models use the same CPU, the BCM2835, but other hardware features differ. The Raspberry Pi Model B+ incorporates several enhancements and new features. Improved power consumption, increased connectivity, and greater IO is among the improvements to this Powerful, small and lightweight ARM-based computer.

2.1.1 Specifications

Chip Broadcom : BCM2835 SoC

Core architecture: ARM11

CPU: 700 MHz Low Power ARM1176JZFS Applications Processor

GPU: Dual Core Video Core IV® Multimedia Co-Processor

Provides Open GL ES 2.0, hardware-accelerated OpenCV, and

1080p30 H.264 high-profile decode

Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure

Memory : 512MB SDRAM

Operating System: Boots from Micro SD card, running a version of the Linux operating system

Dimensions : 85 x 56 x 17mm

Power: Micro USB socket 5V, 2A

Connectors:

Ethernet : 10/100 Base T Ethernet socket

Video Output : HDMI (rev 1.3 & 1.4)

Composite RCA (PAL and NTSC)

Audio Output: 3.5mm jack, HDMI

USB : 4 x USB 2.0 Connector

GPIO Connector : 40-pin 2.54 mm (100 mil) expansion header: 2x20 strip

Providing 27 GPIO pins as well as +3.3 V, +5 V and GND supply lines

Camera Connector: 15-pin MIPI Camera Serial Interface (CSI-2)

Memory Card Slot: Micro SDIO

2.2 General Purpose I/O (GPIO)

GPIO pins can be configured as either general-purpose input, general-purpose output or as one of up to 6 special alternate settings, the functions of which are pin-dependent.

There are 3 GPIO banks on BCM2835.

Each of the 3 banks has its own VDD input pin. On Raspberry Pi, all GPIO banks are supplied from 3.3V. *Connection of a GPIO to a voltage higher than 3.3V will likely destroy the GPIO block within the SoC.*

A selection of pins from Bank 0 is available on the P1 header on Raspberry Pi.

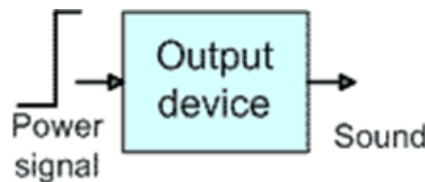


Figure 2-2 Buzzer I/O

2.2.1 Buzzer

The buzzer subsystem produces an audible tone when powered. Buzzers come in a variety of voltages and currents. The power supply for the buzzer (which can be separated from the supply for the rest of the electronics) must provide the voltage needed by the buzzer. Piezo sounders are a type of buzzer. They should not be confused with Piezo transducers – which require an AC input voltage to drive them.



Figure 2-3 Buzzer [17]

2.2.2 Camera

Old webcams may be Full-speed devices. Because these devices transfer a lot of data and incur additional software overhead, reliable operation is not guaranteed.

As a workaround, try to use the camera at a lower resolution. The camera module requires 250mA. For the camera to work, we need to enable the camera in the Raspbian OS.

We go into terminal and type

```
$ sudo raspi-config
```

This gives us options to change many settings. In order to use the Raspberry Pi camera module, you must enable it here. Select the option and proceed to Enable. This will make sure at least 128MB of RAM is dedicated to the GPU.



Figure 2-4 USB Camera

2.3 Setup

All the above-mentioned modules are connected to the raspberry pi using GPIO in the following way. USB camera is used and is connected via USB cable. The GPIO cables are used to connect the buzzer. The buzzer is connected to the IO pad at 23 negative and 25 positive spots. Similarly, male, female pins are taken and connect to the adjacent spots in the IO pad and connect to the 1,3 pins on the raspberry pi.

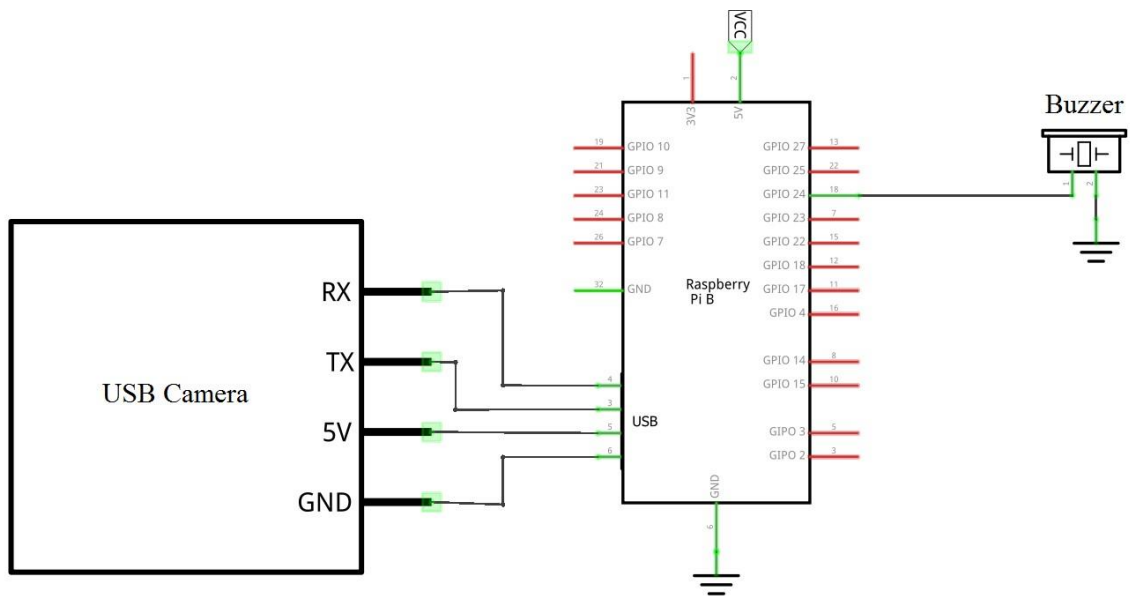


Figure 2-5 IO representation of the raspberry pi setup

2.4 Software Requirements

The following list of software is necessary for the project.

Table 1 Software Requirements

Software	Version
Android studio	1.4
Raspbian OS [18]	4.19
Python	2.7
Java	1.8
Windows	7 or higher
VNC server	1.0 or higher

2.5 Hardware Requirements

Following hardware are needed for the project

Table 2 Hardware Requirements

Hardware	Version
Raspberry pi	3 B+ or 4 B+
USB Camera	720p, 5mp at least
Buzzer	Raspberry pi compatible
Android Phone	Android 5+
PC	-

Apart from these requirements, an internet connection is needed and should be connected to both raspberry pi and the android Phone. Cellular services with SMS services and balance are required for the alert system. These requirements are on the main hardware

and software, detailed requirements of the packages in python are explained and given when code of the project is explained.

2.6 Advantages

Most of the reasons why all the software and hardware used in this project are better than most implementations we have are already explained in the Literature review. Here are most of the advantages of everything used in this project.

The hardware used is minimal and are not very expensive as some might get in other implementations. Most of the hardware needed for the project apart from a computer and a phone (which is least expected to have from everyone) would be around 50\$ USD.

OpenCV and dlib are very flexible to use and for our advantage, already has a pre-built library of things that we use for face and eyes detection.

Android is the most used OS in mobile devices and the requirement is at least android 5.0 which is covered by at least 98% smartphones in the world. This minimal requirement makes the app easily accessible and ready to deploy to anyone.

The Raspberry pi makes sit easy to make this setup mobile and easy to install in a car. If we have a USB accessible dock in the car (which most of the cars have), it is easy to install the hardware.

2.7 Disadvantages

Nothing is perfect in this world and is same with this setup. The implementation in raspberry pi 3 B+ which has just 1 GB of RAM makes the detection a bit harder than usual as we are using dlib which is 30 gigabytes of data. Raspberry pi 3 can only detect around 3 frames per second which is the biggest disadvantages. But fortunately, there is

a solution to this problem, and it is Raspberry pi 4 B+. The new raspberry pi has 4 GB of RAM which can do much better than the one I used to implement. This makes the work so easy. It can increase the frames it can detect in a second by 4 times. So, it will be able to detect around 20 to 25 frames per second. This is very useful to make the detection faster and more accurate.

The only other limitation in this project is the use of a server. The server is reliable as it is regularly maintained by a major company. But to access the server, we need to have a constant Internet connection to both the android phone and raspberry pi. The connect from the android phone can be shared to the raspberry pi and usually, the pi needs to run when the driver is present. This makes this a bit easy as whenever the driver is present, the pi has an internet connection.

Chapter 3

PROJECT OVERVIEW

3.1 Proposed Method

Eye blinking is a very fast operation of a human eye. Everyone's eye blink pattern is different, it differs in closing, opening, squeezing of the eye, etc. it is approximately between 100~300 ms for a blink of an eye. State-of-the-art facial landmark detectors are used in order to detect eyes as they are pre-trained and are widely used in many scenarios. This also helps in comparing this project with previous implementations.

By using the landmarks detected, local eyelids and contours are exploited and eye aspect ratio (EAR) is found. This EAR is compared with the threshold value which varies from person to person based on their average EAR. When the EAR is more than the threshold value, the buzzer will be on and an alarm is triggered.

Python code is implemented to perform all these operations and is implemented in Raspberry pi. An android app controls the functions of the raspberry pi like turn on and off the project, control the volume of the buzzer and to set a contact to send the alert.

3.2 Facial Features and Landmarks

The hardware side of this project is shown and explained later. By using the camera attached to the raspberry pi, continuous real-time face detection is done. After detecting the face, the task is to acquire facial landmarks, for example, eye contours, mouth corners,

nose, eyebrows, etc. Geometrical knowledge is required to locate facial landmarks such as the eye corners and center, the mouth corners and centers.

Our system uses a facial landmark detector that is included in the dlib library. This is mentioned in [10]. Results are achieved faster and effectively with the help of a new algorithm. Dlib's facial landmark detector is pre-trained and is used to find the location of 68(x, y)- coordinates. These coordinates are key coordinates that help to localize certain areas of the face. The 68 coordinates can be visualized in figure 3-1.

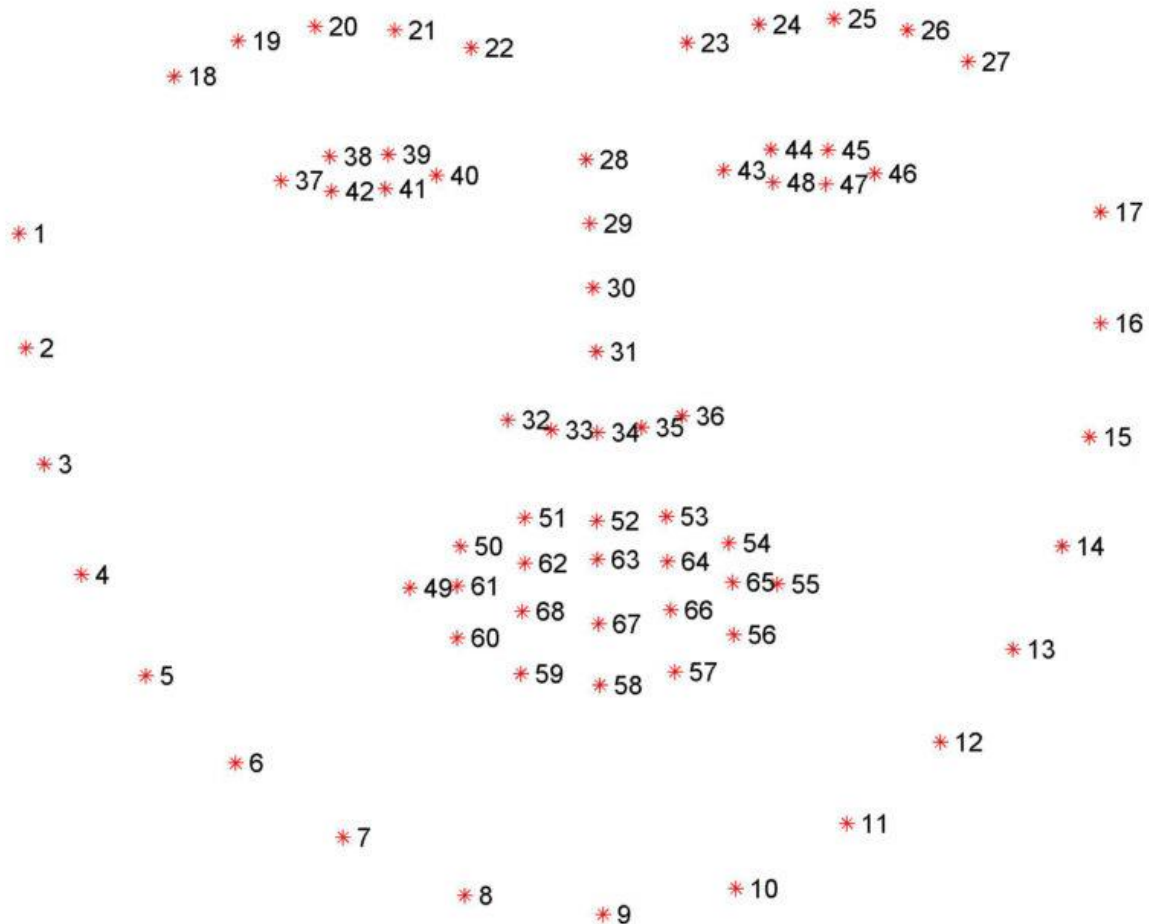


Figure 3-1 representation of 68 Facial Landmarks

If we take a closer look at the figure 3-2, each landmark has a number associated with it. By using this number, we can take a part of the face and process it to use however we want it.

These landmarks do a major part in the face and facial parts detection. They can be used later on to further increase the potential of the project. For example, contours drawn using the landmarks at the lips are used to find the Euclidean distance between the lips. This can be used to find if the person is yawning or not, this can be combined with eye detection in order to make the detection more effective.

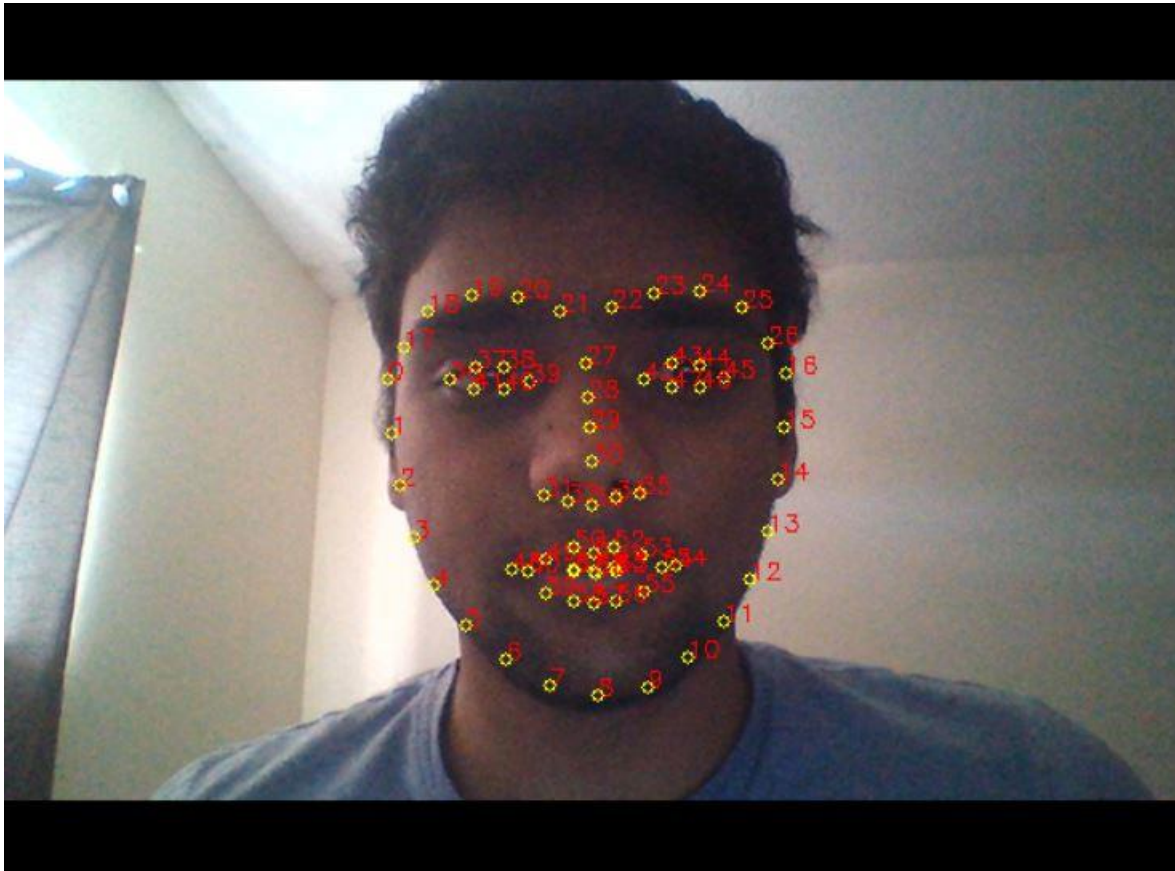


Figure 3-2 Detecting facial Landmarks

If we take a close look at one of the eyes as shown in the figure 3-3, it has 6 landmarks. These 6 landmarks are used to calculate the EAR (eye aspect ratio of an eye). How the EAR is derived is later explained in the code section. Their paper [1] proposed a formula that detects eye blink using the scalar quantity EAR.

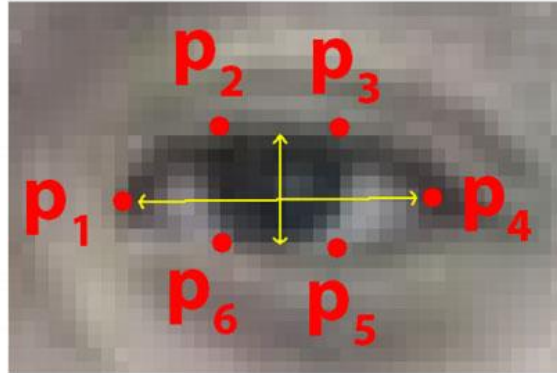


Figure 3-3 eye landmarks

3.3 EAR (eye aspect ratio)

The Eye Aspect Ratio, as the name suggests, is a formula that gives a scalar value of the extent to which the eye is open. The EAR is calculated in each consecutive frame, and when there is a drop in the value of EAR, a blink is detected. The distance between two points, such as p_2 and p_6 are calculated using Euclidean distance which is included in the NumPy library. The Euclidean distance is a popular method to calculate the linear distance between two points. The numerator of the EAR formula in fig 3-4 computes the distance between vertical coordinates and the denominator calculates the distance between the horizontal coordinates. This formula saves us time to find out if the eyes are closed because the conventional method for checking if they are closed relies on further image processing. For the left eye, p_1 to p_6 is from 36 to 41 respectively. Similarly, for the right eye, they are from 42 to 47 respectively.

$$\text{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

Figure 3-4 EAR Formula

We use the same formula and use the landmark positions to determine the EAR of both eyes. Average of EARs of both eyes to get the EAR of the person in that frame. Image 3-5 is a graph that shows the line plot of EAR of several people to the time. We can observe the dip in the EAR value and this is when the person will be drowsy. During this time, all the frames are counted and considered for drowsiness.

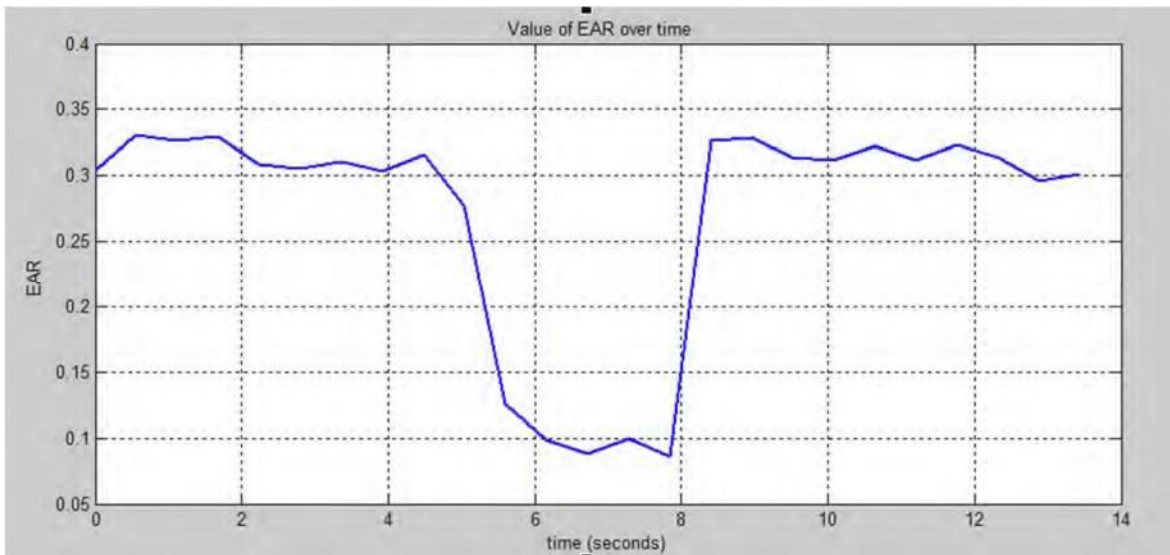


Figure 3-5 EAR vs TIME

3.4 Drowsiness Detection

A single blink is represented in a graph by using EAR in the following figure 3-6. The dip in the line graph is where the person is said to be drowsy. This is very useful in observing the individual's sleep pattern and drowsiness rate. This also can be used to determine the average EAR at the person's regular time. EAR threshold changes from person to person and needs to be set if there is a drastic change in it.

For every frame, EAR detected is compared to the threshold value that we set.

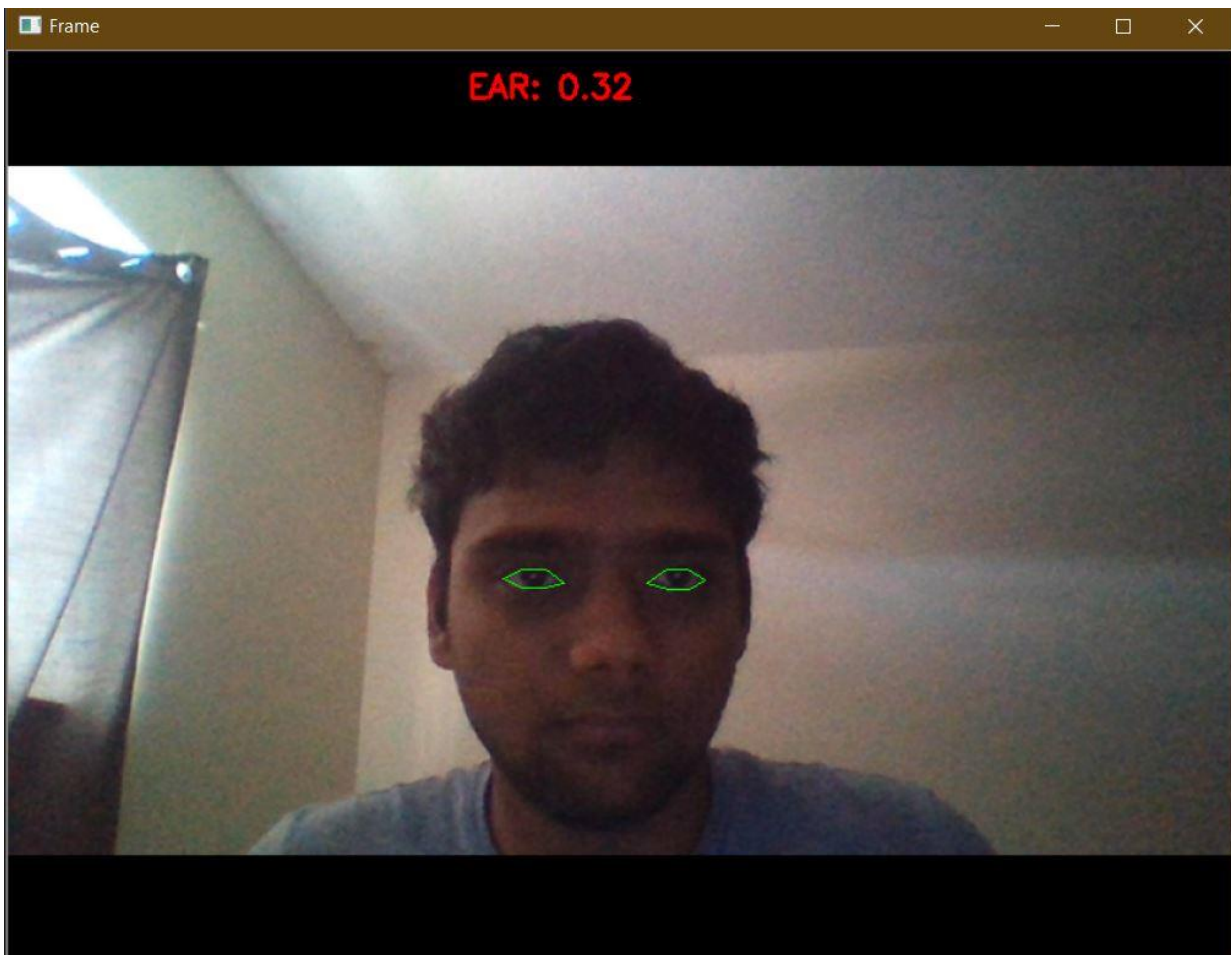


Figure 3-6 EAR detected when the app is launched

If the EAR is less than the threshold value that we set, 0.25 for this experiment, It is taken as drowsiness detected as shown in figure 3-7. An alert is passed from the python program in the raspberry pi to the GPIO pins to trigger the alarm n the buzzer. This will be stopped only when the EAR value reaches more than the threshold again. A Drowsiness alert is shown when the person is drowsy and an alert from the app is sent to a registered mobile as an SMS. This has the name of the person, the location of where he is currently present. The methods used for eye blinking detection, the EAR value calculation and how it can be used to detect the driver's fatigue has also been discussed in this chapter.

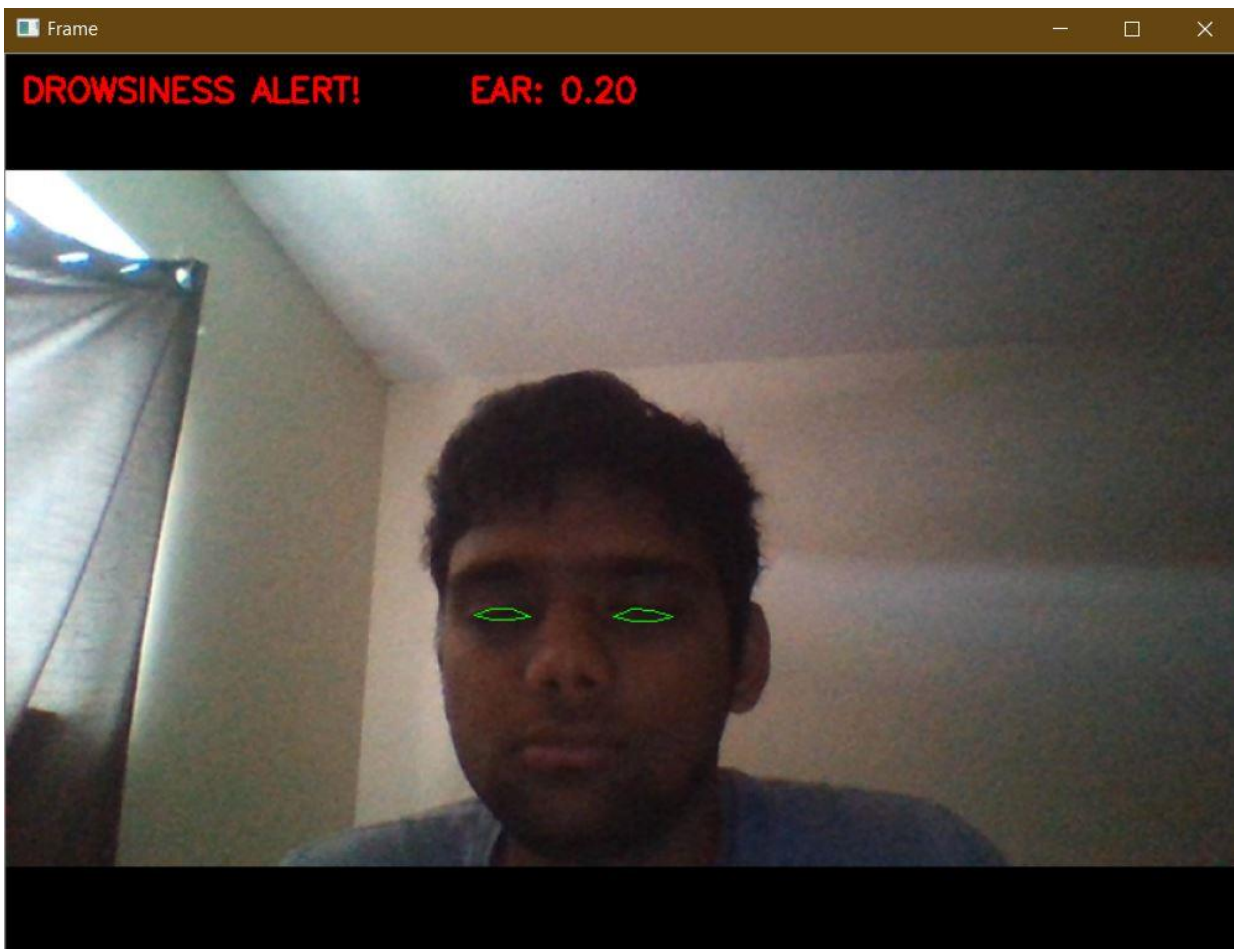


Figure 3-7 Drowsiness alert is shown when detected as drowsy

Chapter 4

ALGORITHM DESIGN AND ANALYSIS

4.1 Flowchart Diagram

Figure 4-1 is a flowchart diagram which shows the functionality of the system or application.

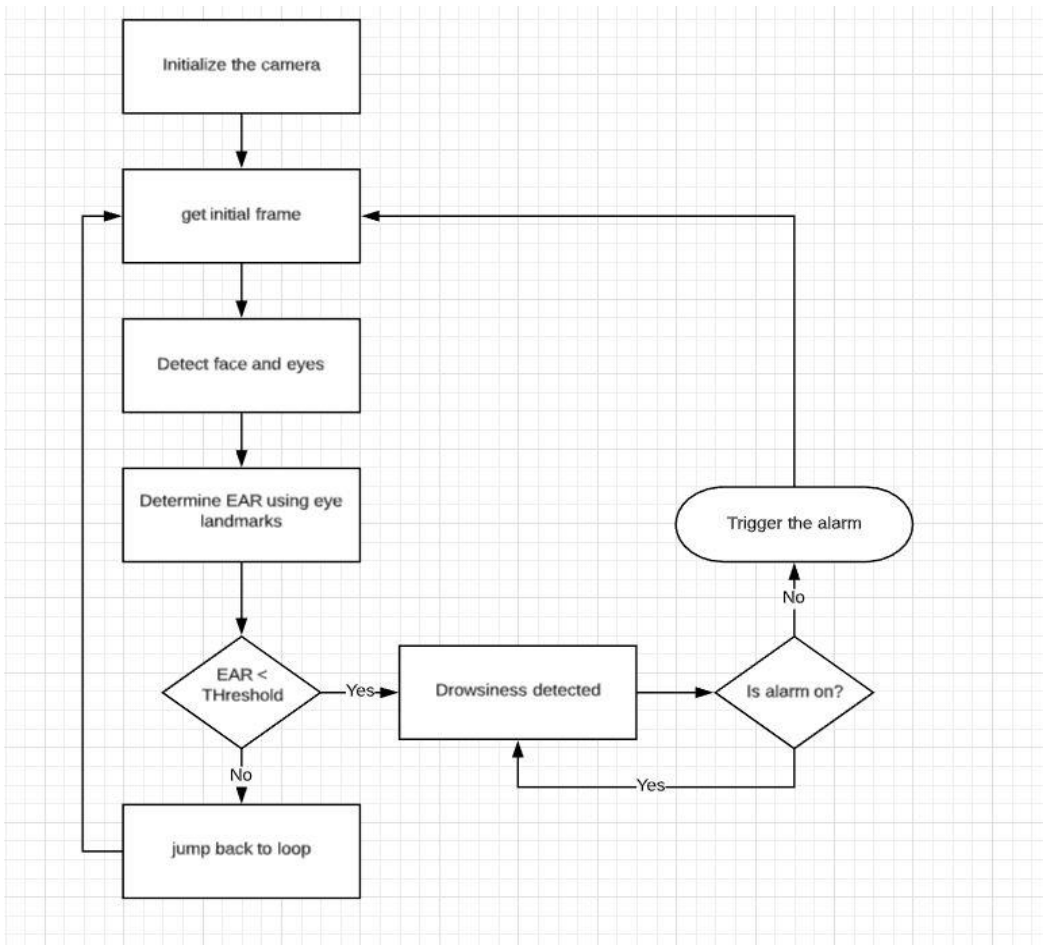


Figure 4-1 Flow Chart Diagram of the System

4.2 Code

Let's look at the python coding that is needed which runs in the raspberry pi. We need to install so many packages before we have the project working. There are so many imports that are necessary for the working of the project. These can be installed through the terminal. For all the above to work, we need to install software components like Python, NumPy, SciPy, OpenCV, Dlib. In figure 4-2 :

Scipy is used for scientific calculations like Euclidean distances

Imutils is used to initialize Video stream and all other parameter related to videos

Thread is taken as we initialize a thread when we receive an alert from android app

Numpy is used for numbers, lists and many other packages in Python

Dlib and Cv2 are used for OpenCV other important packages for face detection.

```
from scipy.spatial import distance as dist
from imutils.video import VideoStream
from imutils.video import FPS
from imutils import face_utils
from threading import Thread
import numpy as np
import argparse
import imutils
import time
import dlib
import cv2
import RPi.GPIO as GPIO
import os
```

Figure 4-2 import packages [B]

Now before installing OpenCV, it is recommended to build the OpenCV library. This build requires around 3-4 hours of time. Dlib library is also needed to work in order to use the 68-face landmark predictor file that contains all the details of the face and eye detection landmarks. The above figure 4-3 contains all the lines of code to start and warm up the IO and set flags to control and initialize the IO parts of the raspberry pi. The BUZ value is used to control the buzzer and flag is set when the driver is drowsy. This flag will be checked later, an alert is made.

```
BUZ=19
flag = 20
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(flag,GPIO.IN)
GPIO.setup(BUZ,GPIO.OUT)
pwm = GPIO.PWM(BUZ, 1000)
pwm.start(0)
```

Figure 4-3 Set IO Values for Raspberry pi [B]

The EAR value which was discussed in the last chapter is found using the Euclidean distance between the landmarks in the eye. From figure 4-4, we can derive that 1, 5 are the extreme ends of the eye. 4 other points are taken, and opposites are considered for the Euclidean distance. Now the average of the 3 values is taken and considered as the EAR.

```
#determine ear
def eye_aspect_ratio(eye):
    A = dist.euclidean(eye[1], eye[5])
    B = dist.euclidean(eye[2], eye[4])
    C = dist.euclidean(eye[0], eye[3])
    ear = (A + B) / (2.0 * C)
    return ear
```

Figure 4-4 EAR calculation [B]

The important values that we initialize. THRESH is the threshold value that we consider which in this case is 0.28 as shown in figure 4-5. This changes for most people as this depends on how much the eyes are closed. For some people eyes are closed more than some other people. This is an average value that works commonly for most, but we need to change if there is a different case which may result in much more false positives.

CONSEC_FRAMES is a value that we use to compare the number of frames the driver is drowsy. We update COUNTER for every frame we find him to be drowsy (with the help of EAR). We keep comparing the counter with consecutive frames which in this case is 8. The low number is taken as the raspberry pi is slow in processing the video stream.

```
# initialise values
THRESH = 0.28
CONSEC_FRAMES = 8
COUNTER = 0
```

Figure 4-5 Initialise values [B]

Here we supply array slice indexes in order to extract the eye regions from the set of facial landmarks. The shape of both eyes are taken by using shape method from the values and this is again used to find the EAR. The values are found and returned by the methods. These are averaged to find the EAR of the person. The Code for the process is given in figure 4-6.

```
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
leftEye = shape[lStart:lEnd]
rightEye = shape[rStart:rEnd]
leftEAR = eye_aspect_ratio(leftEye)
rightEAR = eye_aspect_ratio(rightEye)
ear = (leftEAR + rightEAR) / 2.0
```

Figures 4-6 code to Calculate EAR [B]

Figure 4-7 is the algorithm that is used in the program. If the EAR value is less than the threshold, we check if the counter is more than the threshold of the frame. If It is true, then we consider it as a drowsy scenario and open alert.txt file and put 1 in it. It says that the flag is set to 1 and when buzzer reads this file in another file, it says to ring the buzzer. Another file names ret.txt is also accessed in this. This file has a value of the volume like 30 or 60 which was set in another program.

If this case fails i.e., Ear is more, then it goes to else part and changes the counter to 0 and sets the flag to 0.

```
if ear < EYE_AR_THRESH:
    COUNTER += 1
    if COUNTER >= EYE_AR_CONSEC_FRAMES:
        print("DROWSINESS ALERT!")
        exists = os.path.isfile('alert.txt')
        if exists:
            os.remove('alert.txt')
        f = open('alert.txt', 'a')
        f.write("1")
        f.close()
        f = open('ret.txt', 'r')
        data = f.read()
        f.close()
        pwm.ChangeDutyCycle(float(data))
        cv2.putText(frame, "DROWSINESS ALERT!", (70, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
else:
    COUNTER = 0
```

Figure 4-7 Main algorithm [B] , [16]

The main file has the code to link the raspberry pi files to android app. These functions are used for HTTP request and HTTP post functions. These are used to set different values that we need to set in the android app and access it through a server. To turn on and off the project, we set the flag to 1 or 0 and set put that file in a URL. That URL is accessed using HTTP request. The post is used to post the values of the alarm. It posts the value of alarm if it is on or off. The details of this is taken in python as shown in figure 4-8

```
def request():
    url = "http://ctcorphyd.com/Drowsy/status.php"
    contents = urllib.request.urlopen(url).read()
    #print(contents.decode()[2],contents.decode()[6])
    return contents.decode()[2],contents.decode()[6]

def post(val):
    url = "http://ctcorphyd.com/Drowsy/alertkit.php?sts="+str(val)
    #print(url)
    contents = urllib.request.urlopen(url).read()
```

Figure 4-8 request and post methods [A]

The App design for the project is shown in figure 4-9. This has controls to start or stop the program in raspberry pi , control the volume of the buzzer in 3 different stages and to select contacts to send SMS.



Figure 4-9 App design [D]

A thread is initialized which we use to manipulate the power and alarm sound settings. This if function is used to see if the power is set to 0 or 1. We break the loop in which this function is written when the power is 0, also set the flag to 0. The code for the process is shown in figure 4-10.

```
if(power=='0'):  
    GPIO.output(flag,0)  
    sleep(1)  
    #print("Quitt")  
    break
```

Figure 4-10 code to set power [A]

We have 3 kinds of volume controls in this Program. The below-given picture show one of the functions. We set the volume to one of the 3 options in the dropdown box, then we turn the power on beside the raspberry pi. This sets the parameters power to 1 and volume to 1,2,3 based on the selection. Previously in the figure 4-7, we access a file ret.txt which was created here. The figure 4-11 shows how this is done.

```
if(power=='1'):
    print("Power ON")
    if(volume=='1'):
        exists = os.path.isfile('ret.txt')
        if exists:
            os.remove('ret.txt')
        f = open('ret.txt', 'a')
        f.write('30')
        f.close()
        print("Volume: 30")
```

Figure 4-11 Access app values through files [A]

The android app posts the value of the volume in an URL and it is requested here, once we get it, we put it in a file and send it to the raspberry pi file. In figure 4-12, the options to select the volume in android app is given.

```
public void onItemSelected(AdapterView<?> adapterView,
                           View view, int i, long l) {
    vol=volume.get(i);
    if(!vol.equals("Volume")){

        if(vol.equals("30"))
            vol="1";
        if(vol.equals("60"))
            vol="2";

        if(vol.equals("100"))
            vol="3";

        new SendVolume().execute();
    }
}
```

Figure 4-12 Set volume in the app [C]

The figure 4-13 has the code for the SMS part of the project. First, a string `adrs` is created and has the details of the address of the person which is technically the android phone which can be obtained using the GPS in the phone. Then we initialize SMS manager to create a text message. To get the phone number of the person we are sending we are also accessing the contact of the person. Then we create a toast in order to know that a text message is indeed sent an alert is passed correctly when he is drowsy.

```
if(status.equals("1")) {  
    //Toast.makeText(MainActivity.this, "cnt=s"+cnt, Toast.LENGTH_SHORT).show();  
    if(cnt==0) {  
        String adrs = getAddress(MainActivity.this, latitude, longitude);  
        SmsManager sms = SmsManager.getDefault();  
        sms.sendTextMessage(phoneNumber, null, "vasista is drowsy at "+adrs, null, null);  
        Toast.makeText(MainActivity.this, "send sms"+phoneNumber, Toast.LENGTH_SHORT).show()  
        // doTheAutoRefresh(10000);  
        cnt=1;  
    }  
}
```

Figure 4-13 Code to send SMS with a location in it [C]

Chapter 5

IMPLEMENTATION

5.1 Project Setup

In this section, the setup of the project will be discussed along with how to run the project and what the results we get. In the last chapter, the algorithm is explained (at least the important parts of the code). This explains how we use that algorithm and what do we need to run the setup. Here is the setup (figure 5-1) that is arranged in my house. The screen is not needed when we deploy this into the car. The program is written in python, so we can run the program from terminal or editor that can run python.

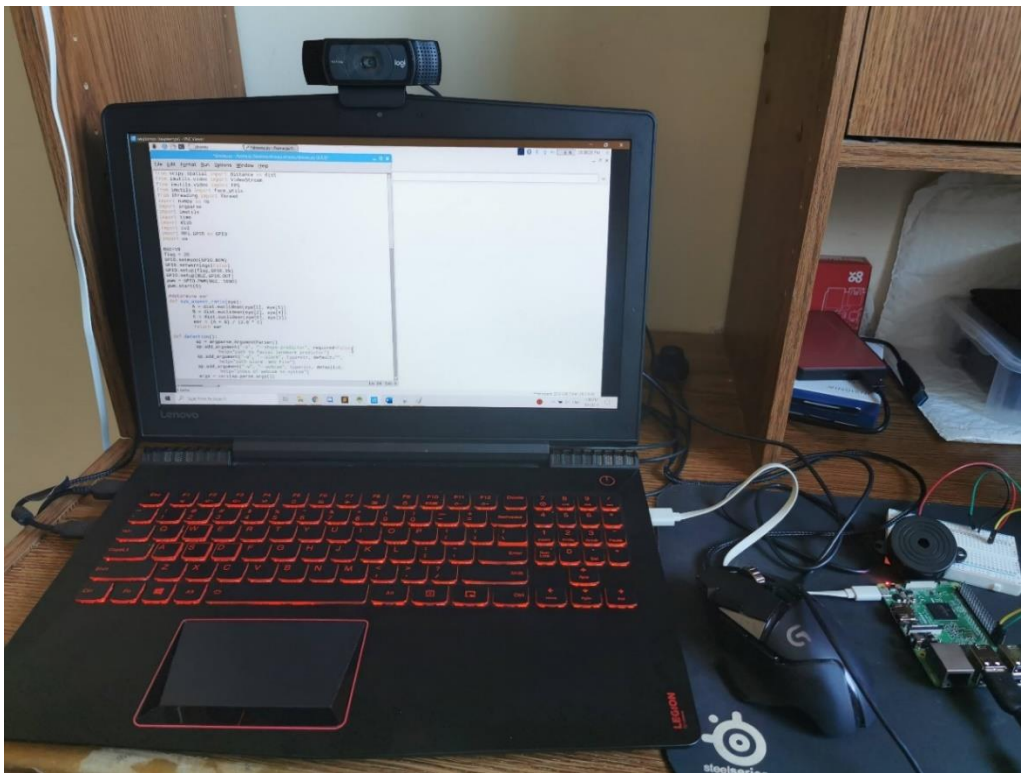


Figure 5-1 Project working setup

5.2 Running the Setup

After everything is connected, let us look at how to run the project. Before running the program, we need to install the app on the smartphone. First, run the android studio version 2.4 or above and open the application that is given. Connect the phone to the computer using a USB cable and turn on developer options along with USB debugging in the smartphone. This is needed to install applications into the android phone.

Once it is detected, it shows up as a device recognized and the device is online. When We try to run the program, it asks to select the device to run the application which looks the figure 5-2 :

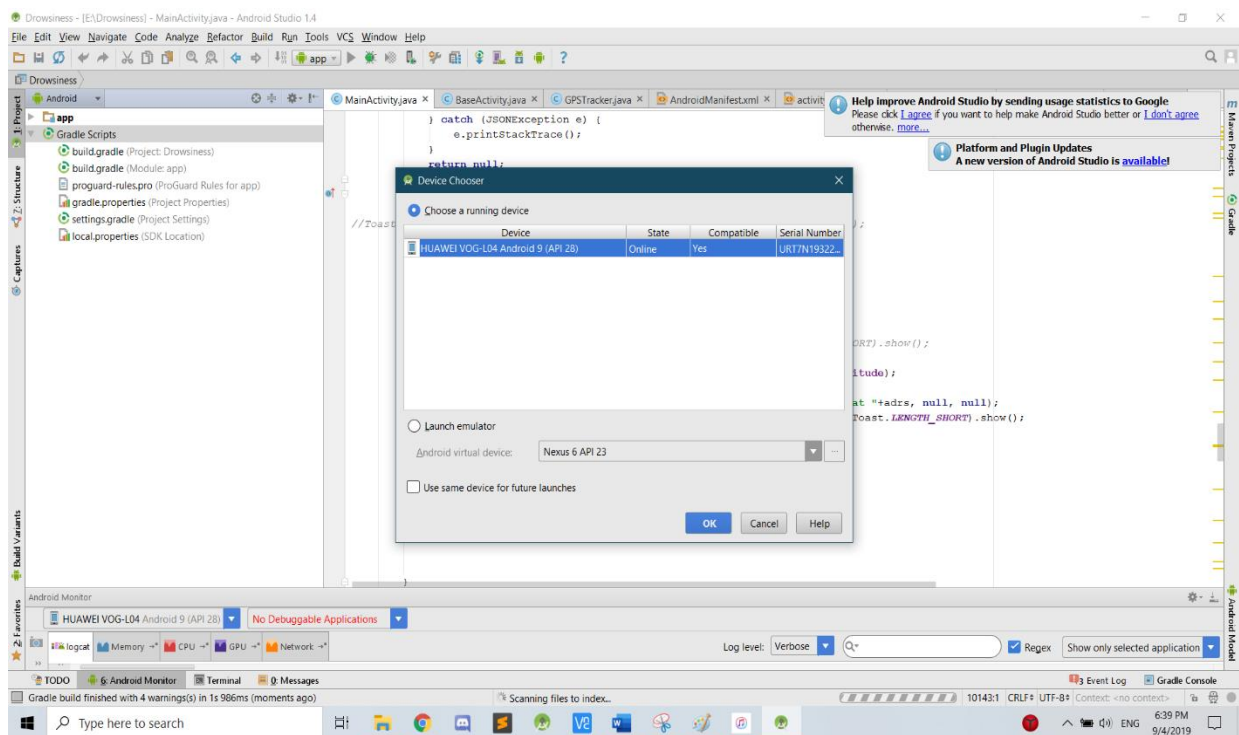


Figure 5-2 Android Studio showing connected phone

After selecting the device, it runs and installs the application in the smartphone. From now on, the computer is not needed to run the application again. Internet connection in the smartphone is a must to run the application. When the application is opened, it shows 3 options

1. Turn on/off the raspberry pi program that we are running
2. Control the volume of the buzzer with 3 options as 30, 60, and 100
3. Contacts list to select a contact to which SMS alert needs to be sent.

These options can and be used once the program is launched in the raspberry pi. Among these options, first, the phone number from contacts need to be set. Before turning the program on, we need to make sure one of the volume options is selected. Figure 5-3 shows the mentioned options.

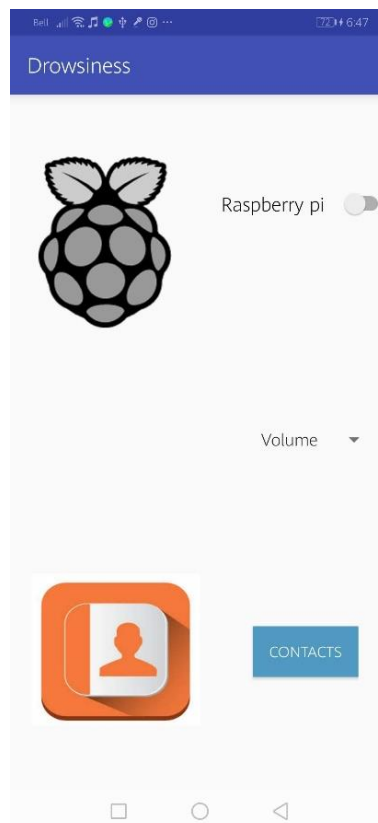


Figure 5-3 App UI when everything is off [D]

Once the program is run in the raspberry pi, it launches a window which looks like the window on the right side in the figure 5-4. It says

Driver Drowsiness detection

Select the power button in the App to start detection.

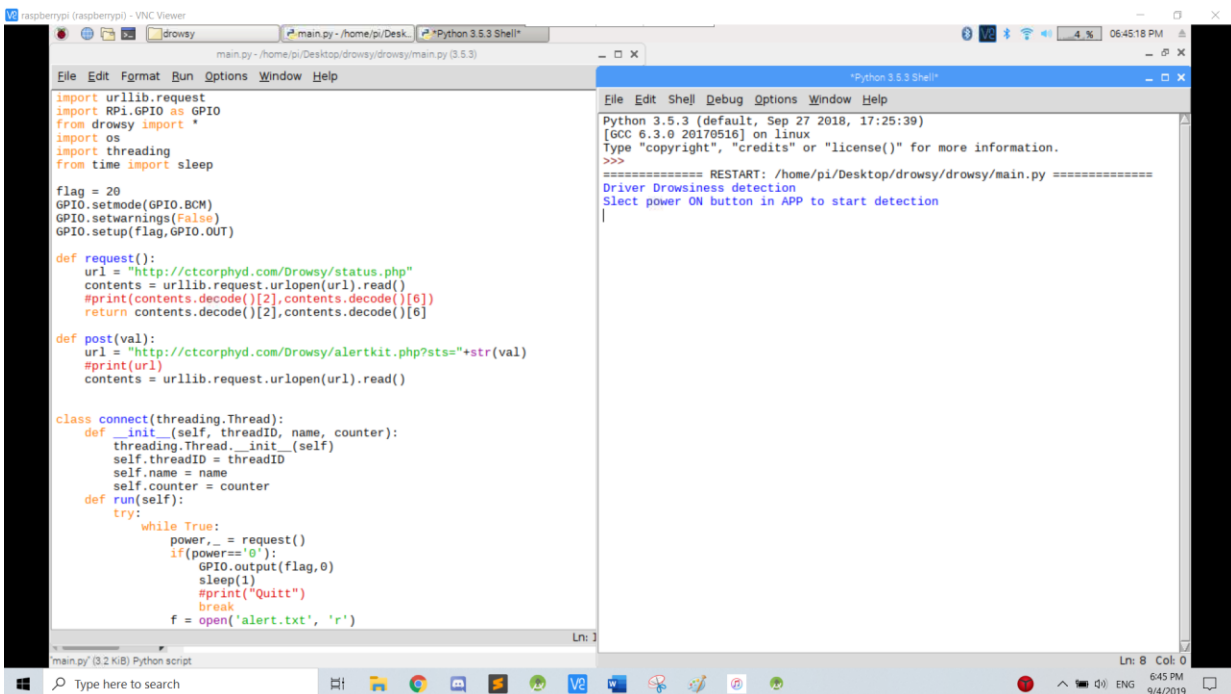


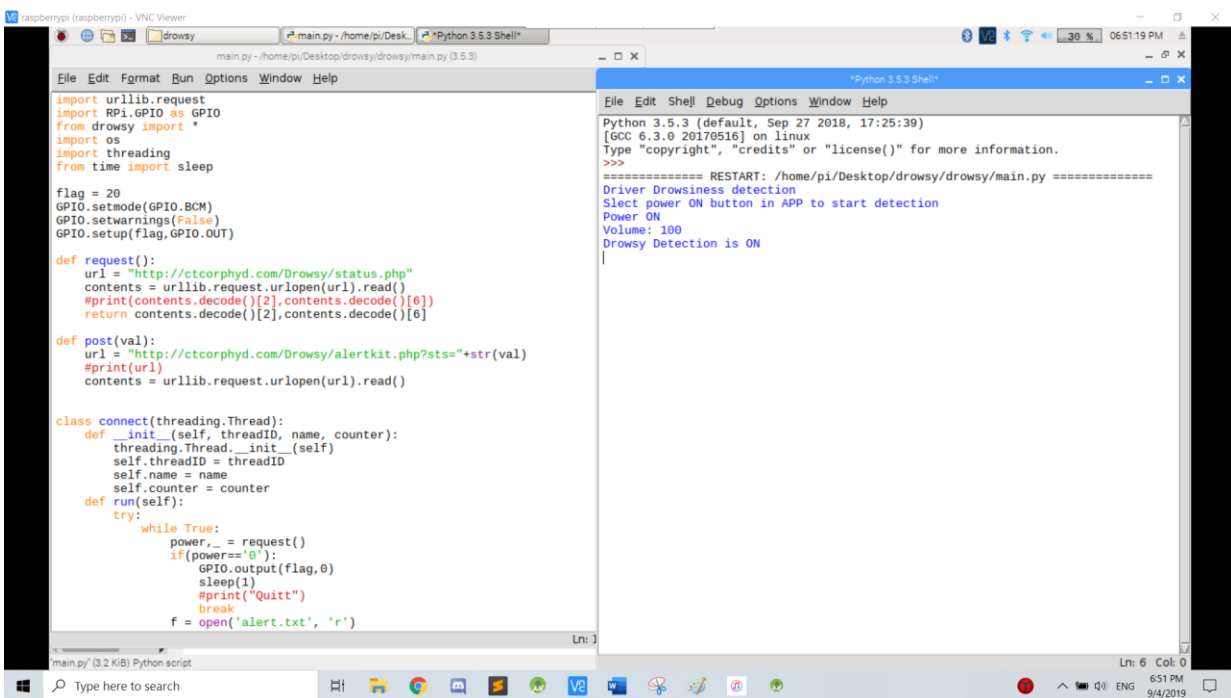
Figure 5-4 Raspberry pi screen showing the program running [A]

Now we open the App to select the options. The order of selecting the options in the app is to select the contact, select the volume and turning the power on. Once all the options are turned on, it looks like the 5-5 figure. Once we select the options in the smartphone, these options are sent to the URLs in the server and accessed by the raspberry pi.



Figure 5-5 App UI showing everything is on [D]

These options are then copied to file and taken and is loaded into the program after which the log shows that the options are taken, and the app started detection which looks like the figure 5-6. The log shows all the values we gave, and raspberry pi started to find the camera and buzzer. It warms up the camera and makes a loud sound using the buzzer as it finds it. It shows that the power is ON, volume is 100 and Drowsy detection is ON. This means the detection has started and a window with the video stream to pop up (we connected It to the monitor, so it happens. This does not happen when the setup is installed in the car).



The screenshot shows a Raspberry Pi desktop environment with a VNC viewer. The main window is a Python 3.5.3 Shell running a script named 'main.py'. The script imports libraries like urllib, RPi.GPIO, and threading, and defines functions for making HTTP requests and controlling a buzzer. The output in the shell window shows the script's execution, including a restart message, driver drowsiness detection status, power ON button selection, volume setting to 100, and confirmation that drowsy detection is ON.

```
File Edit Format Run Options Window Help
main.py - /home/pi/Desktop/drowsy/drowsy/main.py (3.5.3)

import urllib.request
import RPi.GPIO as GPIO
from drowsy import *
import os
import threading
from time import sleep

flag = 20
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(flag,GPIO.OUT)

def request():
    url = "http://ctcorphyd.com/Drowsy/status.php"
    contents = urllib.request.urlopen(url).read()
    #print(contents.decode()[2],contents.decode()[6])
    return contents.decode()[2],contents.decode()[6]

def post(val):
    url = "http://ctcorphyd.com/Drowsy/alertkit.php?sts="+str(val)
    #print(url)
    contents = urllib.request.urlopen(url).read()

class connect(threading.Thread):
    def __init__(self, threadID, name, counter):
        threading.Thread.__init__(self)
        self.threadID = threadID
        self.name = name
        self.counter = counter
    def run(self):
        try:
            while True:
                power,_ = request()
                if(power=='0'):
                    GPIO.output(flag,0)
                    sleep(1)
                    #print("Quitt")
                    break
                f = open('alert.txt', 'r')

Ln: 3

Python 3.5.3 Shell
Python 3.5.3 (default, Sep 27 2018, 17:25:39)
[GCC 6.3.0 20170516] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/pi/Desktop/drowsy/drowsy/main.py =====
>>>
Driver Drowsiness detection
Select power ON button in APP to start detection
Power ON
Volume: 100
Drowsy Detection is ON
|
```

Figure 5-6 Raspberry pi screen showing updated information received from the app [A]

The image 5-7 shows the video stream detecting my face and INFO is given in the log as detection started. This means the detection started and it is warming up camera. EAR for every frame is shown above.

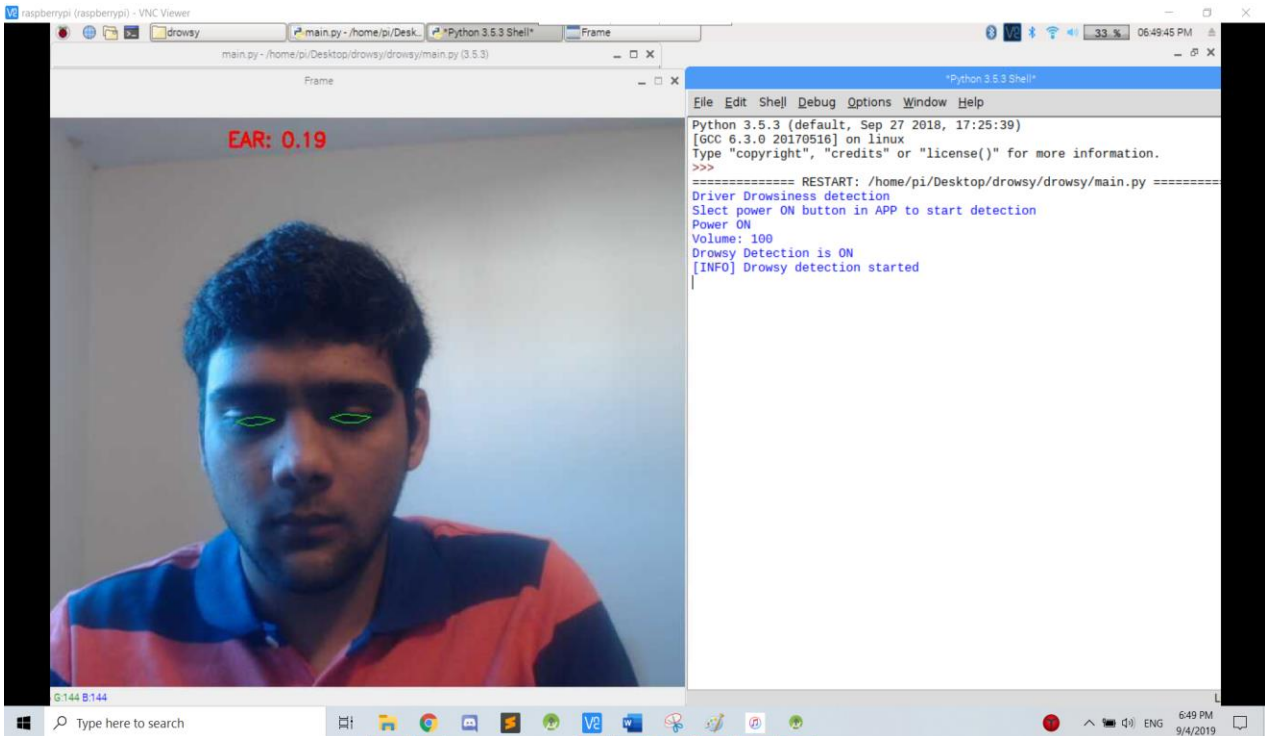
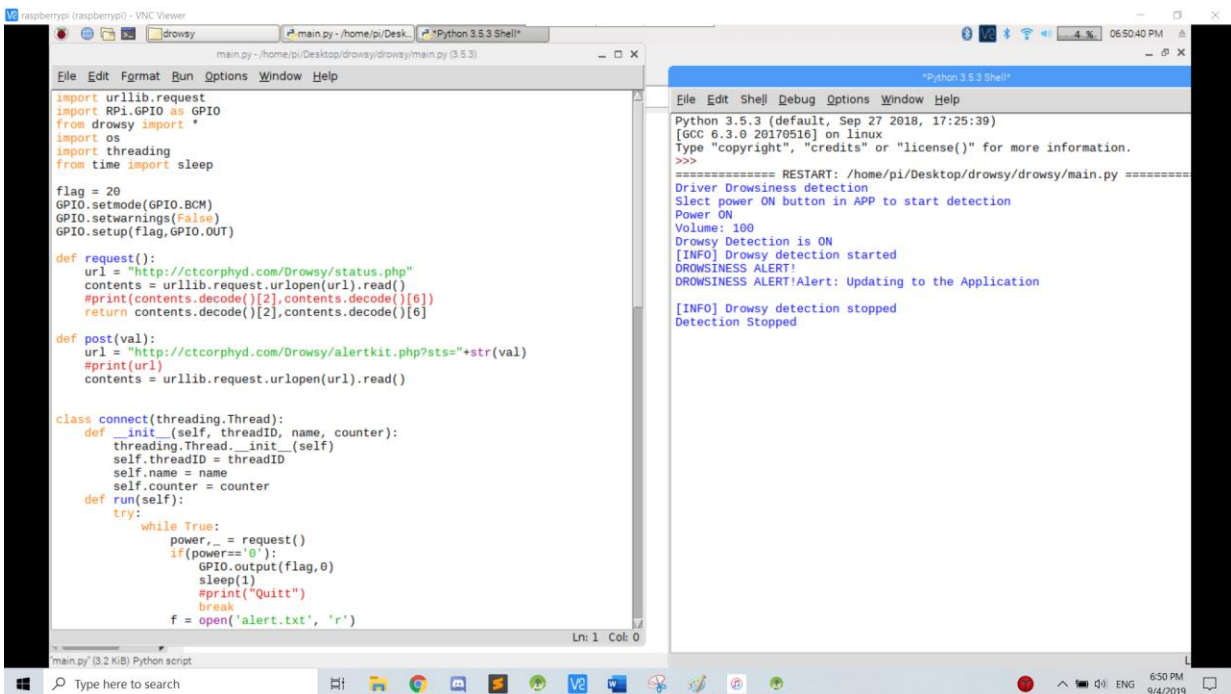


Figure 5-7 Raspberry pi screen showing video stream

The figure 5-8 shows some details after the drowsiness detection is done and the driver is detected as drowsy. It says drowsiness alert and updating the application. This sends an alert to the application i.e., a flag is set saying that the driver is drowsy. When the flag is set to 1, the application changes the value in the url to 1 so that the mobile application can read the value to send an alert to the designated contact.



The screenshot shows a Raspberry Pi desktop environment with two windows. The left window, titled 'main.py - /home/pi/Desktop/drowsy/drowsy/main.py (3.5.3)', displays a Python script. The script imports libraries like urllib, GPIO, os, threading, and time. It defines a 'request()' function to fetch data from 'http://ctcorphyd.com/Drowsy/status.php' and a 'post(val)' function to send data to 'http://ctcorphyd.com/Drowsy/alertkit.php?sts='+str(val)'. A 'connect' class is defined, which inherits from 'threading.Thread' and contains a 'run()' method with a loop that checks a 'power' variable and updates a 'flag' on the GPIO. The right window, titled 'Python 3.5.3 Shell', shows the output of the script. It includes a restart command, a message 'Driver Drowsiness detection', and a series of status updates: 'Select power ON button in APP to start detection', 'Power ON', 'Volume: 100', 'Drowsy Detection is ON', '[INFO] Drowsy detection started', 'DROWSINESS ALERT!', 'DROWSINESS ALERT!Alert: Updating to the Application', '[INFO] Drowsy detection stopped', and 'Detection Stopped'.

```
File Edit Format Run Options Window Help
main.py - /home/pi/Desktop/drowsy/drowsy/main.py (3.5.3)

import urllib.request
import RPi.GPIO as GPIO
from drowsy import *
import os
import threading
from time import sleep

flag = 20
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(flag, GPIO.OUT)

def request():
    url = "http://ctcorphyd.com/Drowsy/status.php"
    contents = urllib.request.urlopen(url).read()
    #print(contents.decode()[2],contents.decode()[6])
    return contents.decode()[2],contents.decode()[6]

def post(val):
    url = "http://ctcorphyd.com/Drowsy/alertkit.php?sts="+str(val)
    #print(url)
    contents = urllib.request.urlopen(url).read()

class connect(threading.Thread):
    def __init__(self, threadID, name, counter):
        threading.Thread.__init__(self)
        self.threadID = threadID
        self.name = name
        self.counter = counter
    def run(self):
        try:
            while True:
                power,_ = request()
                if(power=='0'):
                    GPIO.output(flag,0)
                    sleep(1)
                    #print("Quitt")
                    break
                f = open('alert.txt', 'r')
```

```
File Edit Shell Debug Options Window Help
Python 3.5.3 (default, Sep 27 2018, 17:25:39)
[GCC 6.3.0 20170516] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/pi/Desktop/drowsy/drowsy/main.py =====
Driver Drowsiness detection
Select power ON button in APP to start detection
Power ON
Volume: 100
Drowsy Detection is ON
[INFO] Drowsy detection started
DROWSINESS ALERT!
DROWSINESS ALERT!Alert: Updating to the Application

[INFO] Drowsy detection stopped
Detection Stopped
```

Figure 5-8 Raspberry pi screen showing log details after detection

When the app keeps accessing the server, it gets a message to send the text message to the contact selected. Then it sends an SMS saying that the person is drowsy at the address as shown in the figure 5-9. They can search for the address by copying it in google maps.

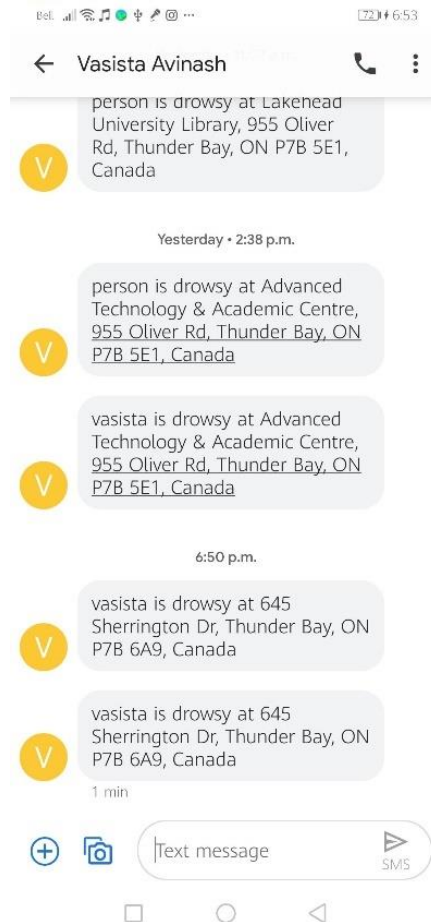


Figure 5-9 Screenshot showing SMS received

It shows the full address of the place the person is drowsy at. This message is sent immediately when an alert is generated. After this, the detection stops once we press the button beside raspberry pi logo. The program keeps running as long as we do not close the application, we just need to press the button in-app again.

Chapter 6

RESULTS & PERFORMANCE ANALYSIS

6.1 EAR analysis

Table 3 EAR of different people

Person	EAR for eyes open	EAR for eyes closed
1	0.39	0.23
2	0.25	0.12
3	0.34	0.16
4	0.33	0.19
5	0.32	0.11
6	0.27	0.12

The EAR values in table 3 are analyzed for different people and with different lighting conditions. The EAR value depends on the eyes of different people which are completely different. As I explained in the above chapters that EAR varies for different people, the following test shows that even when eyes are open some people have EAR low. So, when considering the threshold least possible threshold would be 0.25 as an average for all the people with some exceptional cases like person 2 here. We need to manually change the EAR for these people. As we can see that when the eyes are closed fully the EAR of most of the people are below 0.20. This means 0.25 is a very good value to make sure he/she is not even tipsy.

Graph:

In the graph 6-1, the data in the table 3 is analysed and shown as a visual representation which is easy to understand and it shows how the EAR differs for people when eyes closed and eyes. The least EAR when eyes open does not touch the highest EAR when eyes closed which makes it easy to predict the threshold.

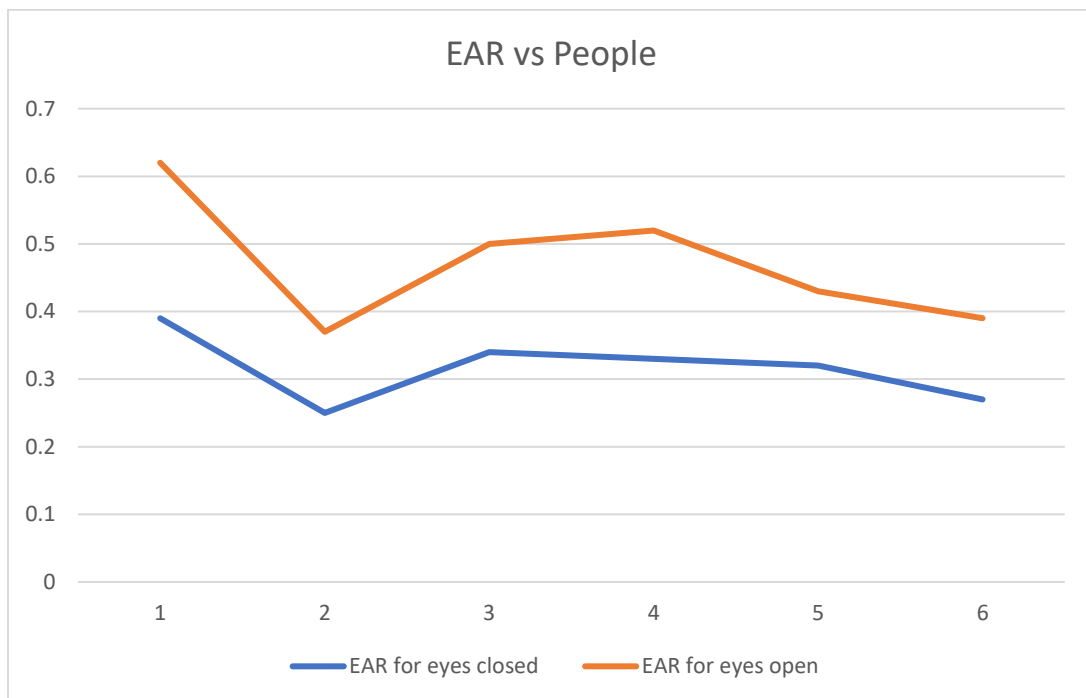
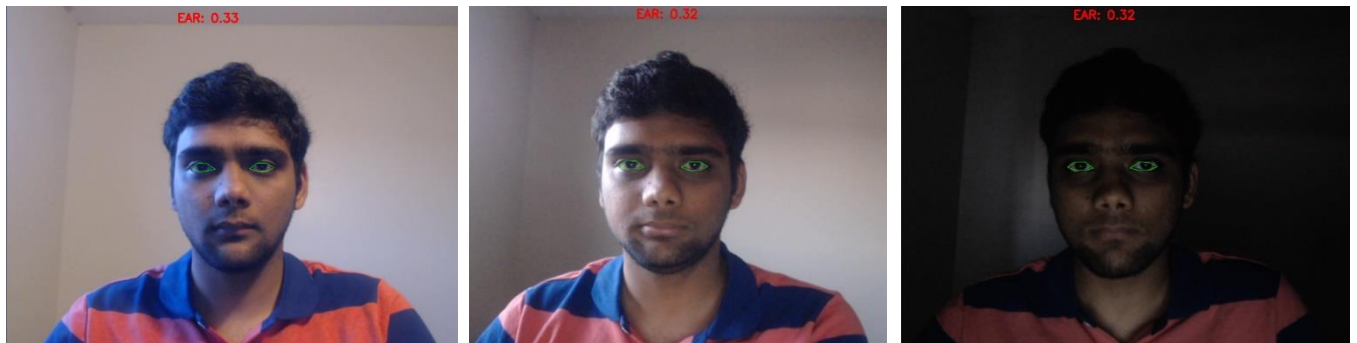


Figure 6-1 EAR vs Different people

For a single person, if we consider different lighting conditions, we can show that the average EAR for a person would be the same for a person in any lighting and at any time in any environment.

Here are 3 different lighting conditions that I created and as we can see there the EAR is 0.33 in the first figure and 0.32 in the other 2 figures in 6-2. This shows that the EAR value does not change drastically for a person. So, once we set a threshold for a person, there is no need to change the threshold for any scenario.



Figures 6-2 3 images shown at different lighting conditions

6.2 Difficulties and Limitations

The hardware limitations like the camera in the extreme dark, processing power of the portable computer i.e., raspberry pi results sometimes in some false-positive results. 9 out of 10 times this setup which I have in my home detects that the person is drowsy correctly. So, this project is 90% accurate but, It is only bound to the hardware limitation. This will be 100% accurate if the hardware capabilities in the future increase and dlib and OpenCV are easy to run.

Chapter 7

CONCLUSION AND FUTURE WORK

6.1 Conclusion

There is a need for a system to detect and wake up the drivers who are drowsy that results in a lot of accidents. This project shows an effective way to detect the drowsiness in a driver. EAR is very effective and easy to detect the eyes and it has high accuracy. When compared to the industry standard HOG cascades, Haar cascade which is what the shape predictor that I used in this project is very fast and used more eye image database to construct facial landmarks. Raspberry pi is not the fastest device there is but, it is very versatile, mobile and easy to install which makes this project very easy to setup. The mobile app is a bonus to the project as it is a UI that is easy for common people to understand. This makes everyone control the project in the raspberry pi to be controlled easy.

6.2 Future Work

A small-scale upgrade can be done in the part of the hardware. Upgrading this setup to a raspberry pi 4 can boost the processing times by double. To do all the hardware side of upgrades is expensive which is quite opposite of what the theme of the project is. But even after upgrading these, the cost will not be more than what other conventional setups doing the same thing has.

IR camera can also be attached to the raspberry pi to increase the detection in the night which is more effective for the drivers in the night. There is so much potential for the

android app to develop. With the implementations getting easy in android, the whole project could be done in android itself. That makes all the work so easy and everything needed comes in a single package as a smartphone.

Even with the project as it is, the app could be expanded with another module where emergency services are contacted. With the help of raspberry pi connected to the vehicle, when the drowsiness is detected, brakes for the vehicle can be applied. The alert that we used to contact with the android app to send SMS can be used to contact emergency services if raspberry pi detects a collision. This also helps in easy contact with emergency services once an accident happens. This makes this project not only accident avoidance but also something can be done when there is an accident.

With the scientific advancements, more equipment is getting cheaper and cars are getting automated with an OS installed in the cars for different purposes. This system can be deployed wherever python can run and there is an internet connection. That will be a huge leap as it directly integrates this project into the cars and helps a lot.

REFERENCES

1. Real-Time Eye Blink Detection using Facial Landmarks Soukupová and Čech, Faculty of Electrical Engineering, Czech Technical University in Prague
2. L. M. Bergasa, J. Nuevo, M. A. Sotelo, and M. Vazquez. Real-time system for monitoring driver vigilance. In IEEE Intelligent Vehicles Symposium, 2004.
3. Medicton group. The system I4Control. [http:// www.i4tracking.cz/](http://www.i4tracking.cz/). 1
4. J. Cech, V. Franc, and J. Matas. A 3D approach to facial landmarks: Detection, refinement, and tracking. In Proc. International Conference on Pattern Recognition, 2014.
5. A. Asthana, S. Zafeoriou, S. Cheng, and M. Pantic. Incremental face alignment in the wild. In Conference on Computer Vision and Pattern Recognition, 2014. 1, 2, 3, 4, 5, 7
6. X. Xiong and F. De la Torre. Supervised descent methods and its applications to face alignment. In Proc. CVPR, 2013. 2, 3, 4, 5
7. F. M. Sukno, S.-K. Pavani, C. Butakoff, and A. F. Frangi. Automatic assessment of eye blinking patterns through statistical shape models. In ICVS, 2009. 1, 2
8. S. Ren, X. Cao, Y. Wei, and J. Sun. Face alignment at 3000 fps via regressing local binary features. In Proc. CVPR, 2014. 2
9. 4. C. Boke, A. Shetty, A. Kadam, S. Jadhav, S. Pakhmode, S. Barahate, et al., “Smart Band for Drowsiness Detection to Prevent Accidents, vol. 5, issue 1, pp. 1107-1114,
10. One Millisecond Face Alignment with an Ensemble of Regression Trees, Kazemi and Sullivan, 2014
11. The Daily Star. ‘Road Accidents: Sharp rise in fatalities’. (2017). Available: <https://www.thedailystar.net/backpage/road-accidents-sharp-rise-fatalities1426999>[Accessed 16 Apr. 2018].

12. M. Divjak and H. Bischof. Eye blink based fatigue detection for prevention of computer vision syndrome. In IAPR Conference on Machine Vision Applications, 2009. 1
13. T. Drutarovsky and A. Fogelton. Eye blink detection using variance of motion vectors. In Computer Vision - ECCV Workshops. 2014. 1, 2, 5, 6, 7
14. Z. Yan, L. Hu, H. Chen, and F. Lu. Computer vision syndrome: A widely spreading but largely unknown epidemic among computer users. Computers in Human Behaviour, (24):2026–2042, 2008. 1
15. D. Torricelli, M. Goffredo, S. Conforto, and M. Schmid. An adaptive blink detector to initialize and update a view-based remote eye gaze tracking system in a natural scenario. Pattern Recogn. Lett., 30(12):1144–1150, Sept. 2009. 1
16. <https://www.pyimagesearch.com/2017/10/23/raspberry-pi-facial-landmarks-drowsiness-detection-with-opencv-and-dlib/>
17. ModMyPi | Raspberry Pi Camera Board - Night Vision & Adjustable-Focus Lens (5MP). (n.d.). Retrieved from <https://www.modmypi.com/raspberrypi/camera/camera-boards/raspberry-pi-night-vision-camera>[Accessed: 26-February- 2018]
18. Best Operating Systems for Raspberry Pi 2 & 3 | Top 10 OS List. (2017, September 20). Retrieved from <https://www.raspberrypistarterkits.com/products/operatingsystems-raspberry-pi/>[Accessed: 26-February-2018]

APPENDIXES

Appendix A: MAIN.PY (RASPBERRY PI)

```
import urllib.request
import RPi.GPIO as GPIO
from drowsy import *
import os
import threading
from time import sleep

flag = 20
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(flag,GPIO.OUT)

def request():
    url = "http://ctcorphyd.com/Drowsy/status.php"
    contents = urllib.request.urlopen(url).read()
    #print(contents.decode()[2],contents.decode()[6])
    return contents.decode()[2],contents.decode()[6]

def post(val):
    url = "http://ctcorphyd.com/Drowsy/alertkit.php?sts="+str(val)
    #print(url)
    contents = urllib.request.urlopen(url).read()

class connect(threading.Thread):
    def __init__(self, threadID, name, counter):
        threading.Thread.__init__(self)
        self.threadID = threadID
        self.name = name
        self.counter = counter
    def run(self):
        try:
```

```

while True:
    power,_ = request()
    if(power=='0'):
        GPIO.output(flag,0)
        sleep(1)
        #print("Quitt")
        break
    f = open('alert.txt', 'r')
    val=f.read()
    #print("val: ",str(val))
    f.close()
    if str(val)=='1':
        print("Alert: Updating to the Application")
        post(1)
    else:
        post(0)
except KeyboardInterrupt:
    print("Quit")

```

```

if __name__ == "__main__":
    try:
        print("Driver Drowsiness detection")
        print("Slect power ON button in APP to start detection")
        while True:
            power,volume = request()
            if(power=='1'):
                print("Power ON")
                if(volume=='1'):
                    exists = os.path.isfile('ret.txt')
                    if exists:
                        os.remove('ret.txt')
                    f = open('ret.txt', 'a')
                    f.write('30')
                    f.close()
                    print("Volume: 30")
                elif(volume=='2'):
                    exists = os.path.isfile('ret.txt')
                    if exists:
                        os.remove('ret.txt')
                    f = open('ret.txt', 'a')
                    f.write('60')
                    f.close()
                    print("Volume: 60")
                elif(volume=='3'):
                    exists = os.path.isfile('ret.txt')
                    if exists:

```

```

        os.remove('ret.txt')
        f = open('ret.txt', 'a')
        f.write('100')
        f.close()
        print("Volume: 100")
        print("Drowsy Detection is ON")
        GPIO.output(flag,1)
        thread1 = connect(1, "check", 1)
        thread1.start()
        #thread1.join()
        detection()
        t2 = threading.Thread(target=connect)
        t2.start()
        t2.join()
        print("Detection Stopped")

except KeyboardInterrupt:
    print("Closing Application")

```

Appendix B: DROWSY.PY (RASPBERRY PI)

```
from scipy.spatial import distance as dist
from imutils.video import VideoStream
from imutils.video import FPS
from imutils import face_utils
from threading import Thread
import numpy as np
import argparse
import imutils
import time
import dlib
import cv2
import RPi.GPIO as GPIO
import os

BUZ=19
flag = 20
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(flag,GPIO.IN)
GPIO.setup(BUZ,GPIO.OUT)
pwm = GPIO.PWM(BUZ, 1000)
pwm.start(0)

#determine ear
def eye_aspect_ratio(eye):
    A = dist.euclidean(eye[1], eye[5])
    B = dist.euclidean(eye[2], eye[4])
    C = dist.euclidean(eye[0], eye[3])
    ear = (A + B) / (2.0 * C)
    return ear

def detection():
    ap = argparse.ArgumentParser()
    ap.add_argument("-p", "--shape-predictor", required=False,
                    help="path to facial landmark predictor")
    ap.add_argument("-a", "--alarm", type=str, default="",
                    help="path alarm .WAV file")
    ap.add_argument("-w", "--webcam", type=int, default=0,
                    help="index of webcam on system")
    args = vars(ap.parse_args())
```



```

# initialise values
EYE_AR_THRESH = 0.28
EYE_AR_CONSEC_FRAMES = 8
COUNTER = 0

PREDICTOR_PATH = "shape_predictor_68_face_landmarks.dat"
detector = dlib.get_frontal_face_detector()

predictor = dlib.shape_predictor(PREDICTOR_PATH)

(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]

vs = VideoStream(src=args["webcam"]).start()
fps = FPS().start()
time.sleep(1.0)
pwm.ChangeDutyCycle(100)
time.sleep(1)
pwm.ChangeDutyCycle(0)
print("[INFO] Drowsy detection started")
while GPIO.input(flag)==1 c :
#while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=800, height=600)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    rects = detector(gray, 0)
    for rect in rects:
        shape = predictor(gray, rect)
        shape = face_utils.shape_to_np(shape)
        leftEye = shape[lStart:lEnd]
        rightEye = shape[rStart:rEnd]
        leftEAR = eye_aspect_ratio(leftEye)
        rightEAR = eye_aspect_ratio(rightEye)
        ear = (leftEAR + rightEAR) / 2.0
        leftEyeHull = cv2.convexHull(leftEye)
        rightEyeHull = cv2.convexHull(rightEye)
        cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
        cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)

    if ear < EYE_AR_THRESH:
        COUNTER += 1
        if COUNTER >= EYE_AR_CONSEC_FRAMES:
            print("DROWSINESS ALERT!")
            exists = os.path.isfile('alert.txt')
            if exists:
                os.remove('alert.txt')
            f = open('alert.txt', 'a')
            f.write("1")

```

```

        f.close()
        f = open('ret.txt', 'r')
        data = f.read()
        f.close()
        pwm.ChangeDutyCycle(float(data))
        cv2.putText(frame, "DROWSINESS ALERT!", (70, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
    else:
        COUNTER = 0
        exists = os.path.isfile('alert.txt')
        if exists:
            os.remove('alert.txt')
        f = open('alert.txt', 'w')
        f.close()
        pwm.ChangeDutyCycle(0)
        cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF
    if key == ord("q"):
        break
vs.stop()
print("[INFO] Drowsy detection stopped")
cv2.destroyAllWindows()

#detection()

```

Appendix C: MAINACTIVITY.JAVA (ANDROID)

```
package ct.drowsiness;

import android.Manifest;
import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.graphics.Bitmap;
import android.location.Address;
import android.location.Geocoder;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Handler;
import android.provider.ContactsContract;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.SwitchCompat;
import android.support.v7.widget.Toolbar;
import android.telephony.gsm.SmsManager;
import android.util.Log;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Spinner;
import android.widget.TextView;
```

```

import android.widget.Toast;
import android.widget.ToggleButton;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;

public class MainActivity extends BaseActivity {
    int sts1=0,sts2=0;
    private ProgressDialog pDialog, pDialog1, pDialog2, pDialog3;
    ArrayList<String> volume = new ArrayList<String>();
    JSONParser jsonParser = new JSONParser();
    private static final String TAG_SUCCESS = "success";
    private static String url_onoff = "http://www.ctcorphyd.com/Drowsy/onoff.php";
    private static final String url_status= "http://www.ctcorphyd.com/Drowsy/alertsts.php";
    private static String url_volume = "http://www.ctcorphyd.com/Drowsy/volume.php";
    private static String url_all = "http://www.ctcorphyd.com/Drowsy/sendall.php";
    double latitude;
    double longitude;
    int cnt=0;
    SwitchCompat switchCompat;
    String sts,vol,status="0";
    ImageView imageView1;
    TextView tv;
    String phoneNumber="",emailAddress="";
    private final Handler handler = new Handler();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Spinner spinner = (Spinner) findViewById(R.id.spinner1);
        switchCompat=(SwitchCompat)findViewById(R.id.switchButton);
        ImageView iv = (ImageView) findViewById(R.id.imageView);
        tv = (TextView) findViewById(R.id.tv);
        isSmsPermissionGranted();
        requestReadAndSendSmsPermission();

        GPSTracker gps = new GPSTracker(MainActivity.this);
        // check if GPS enabled
        if(gps.canGetLocation()){
            latitude = gps.getLatitude();

```

```

        longitude = gps.getLongitude();
        //location=gps.getLocation();
    }else{
        gps.showSettingsAlert();
    }
    new SendAll().execute();
    doTheAutoRefresh();

switchCompat.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {

        if (isChecked) {
            ImageView iv = (ImageView) findViewById(R.id.imageView);
            iv.setBackgroundResource(R.drawable.raspberryon);
            sts= "1";

            //Snackbar.make(buttonView, " ON", Snackbar.LENGTH_LONG).setAction("ACTION",
null).show();

        } else {
            ImageView iv = (ImageView) findViewById(R.id.imageView);
            iv.setBackgroundResource(R.drawable.raspberryoff);
            // Snackbar.make(buttonView, "OFF", Snackbar.LENGTH_LONG).setAction("ACTION",
null).show();
            sts = "0";
        }
        new Sending().execute();

    }
});

volume.add("Volume");
volume.add("30");
volume.add("60");
volume.add("100");

// Create the ArrayAdapter
ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>(MainActivity.this
    ,android.R.layout.simple_spinner_item,volume);

// Set the Adapter
spinner.setAdapter(arrayAdapter);

```

```

spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {

    public void onItemSelected(AdapterView<?> adapterView,
                               View view, int i, long l) {
        vol=volume.get(i);
        if(!vol.equals("Volume")){

            if(vol.equals("30"))
                vol="1";
            if(vol.equals("60"))
                vol="2";

            if(vol.equals("100"))
                vol="3";

            new SendVolume().execute();
        }

    }
    public void onNothingSelected(AdapterView<?> arg0) {
    }
});

}

```

```

class Sending extends AsyncTask<String, String, String> {

    /**
     * Before starting background thread Show Progress Dialog
     */
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        pDialog = new ProgressDialog(MainActivity.this);
        //pDialog.setMessage("Loding...");
        pDialog.setIndeterminate(false);
        pDialog.setCancelable(true);
        // pDialog.show();
    }

    protected String doInBackground(String... args) {

        // Building Parameters
        List<NameValuePair> params = new ArrayList<NameValuePair>();
    }
}

```

```

        params.add(new BasicNameValuePair("sts",sts));
        // getting JSON Object
        // Note that create product url accepts POST method
        JSONObject json = jsonParser.makeHttpRequest(url_onoff,
            "POST", params);

        // check log cat for response
        Log.d("Response for Register", json.toString());

        // check for success tag
        try {
            int success = json.getInt(TAG_SUCCESS);

            if (success == 2) {

                sts1=2;
            }

            if (success == 1) {
                sts1=1;
            } else {
                // failed to create product
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }

        return null;
    }

    /**
     * After completing background task Dismiss the progress dialog
     * **/
    protected void onPostExecute(String file_url) {
        // dismiss the dialog once done
        if(sts1==1) {

            // Toast.makeText(getApplicationContext(), "Inserted", Toast.LENGTH_LONG).show();
        }
        pDialog.dismiss();
    }
}

class SendVolume extends AsyncTask<String, String, String> {
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }
}

```

```

        pDialog1 = new ProgressDialog(MainActivity.this);
        pDialog1.setIndeterminate(false);
        pDialog1.setCancelable(true);
    }

    protected String doInBackground(String... args) {

        List<NameValuePair> params = new ArrayList<NameValuePair>();
        params.add(new BasicNameValuePair("volume",vol));
        JSONObject json = jsonParser.makeHttpRequest(url_volume,
            "POST", params);
        Log.d("Response for Register", json.toString());
        try {
            int success = json.getInt(TAG_SUCCESS);
            if (success == 1) {
                sts2=1;
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return null;
    }

    protected void onPostExecute(String file_url) {
        if(sts2==1) {
            //Toast.makeText(getApplicationContext(), "Inserted", Toast.LENGTH_LONG).show();
        }
        pDialog1.dismiss();
    }
}

```

```

class SendAll extends AsyncTask<String, String, String> {
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }

    protected String doInBackground(String... args) {

        List<NameValuePair> params = new ArrayList<NameValuePair>();
        JSONObject json = jsonParser.makeHttpRequest(url_all,
            "POST", params);
        try {
            int success = json.getInt(TAG_SUCCESS);
            if (success == 1) {
                sts2=1;
            }
        }
    }
}

```



```

        } catch (JSONException e) {
            e.printStackTrace();
        }
        return null;
    }
    protected void onPostExecute(String file_url) {
        if(sts2==1) {
            // Toast.makeText(MainActivity.this,"sendall",Toast.LENGTH_SHORT).show();
            new MainActivity.Status().execute();
        }
    }
}

class Status extends AsyncTask<String, String, String> {
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }

    protected String doInBackground(String... args) {

        List<NameValuePair> params = new ArrayList<NameValuePair>();
        JSONObject json = jsonParser.makeHttpRequest(url_status,
            "POST", params);
        try {
            int success = json.getInt(TAG_SUCCESS);
            if (success == 1) {
                sts2=1;
                status=json.getString("alert");
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return null;
    }
    protected void onPostExecute(String file_url) {
        if(sts2==1) {
//Toast.makeText(MainActivity.this,"Status=s"+status,Toast.LENGTH_SHORT).show();
            GPSTracker gps = new GPSTracker(MainActivity.this);
            if(gps.canGetLocation()){
                latitude = gps.getLatitude();
                longitude = gps.getLongitude();
            }

            if(status.equals("1")){
                //Toast.makeText(MainActivity.this,"cnt=s"+cnt,Toast.LENGTH_SHORT).show();
                if(cnt==0) {

```

```

        String adrs = getAddress(MainActivity.this, latitude, longitude);
        SmsManager sms = SmsManager.getDefault();
        sms.sendTextMessage(phoneNumber, null, "vasista is drowsy at "+adrs, null, null);
        Toast.makeText(MainActivity.this, "send sms"+phoneNumber,
Toast.LENGTH_SHORT).show();
        // doTheAutoRefresh(10000);
        cnt=1;
    }
    }else{
        cnt=0;
    }

    }

    }

}

private void doTheAutoRefresh() {
    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            doRefreshingStuff(); // this is where you put your refresh code
            doTheAutoRefresh();
        }
    }, 1000);
}

public void doRefreshingStuff(){
    // Toast.makeText(this,"k",Toast.LENGTH_LONG).show();
    new Status().execute();
}

public String getAddress(Context context, double lat, double lng) {
    Geocoder geocoder = new Geocoder(context, Locale.getDefault());
    try {
        List<Address> addresses = geocoder.getFromLocation(lat, lng, 1);
        Address obj = addresses.get(0);

        String add =obj.getAddressLine(0);
        //obj.getSubLocality();// perticular area like ameerpet.
        //add = add + "\n" + obj.getCountryName();
        // add = add + "\n" + obj.getCountryCode();
        //add = add + "\n" + obj.getAdminArea();
        //add = add + "\n" + obj.getPostalCode();
    }
}

```

```

        //add = add + "\n" + obj.getSubAdminArea();
        // add = add + "\n" + obj.getAddressLine(1);
        //add = add + "\n" + obj.getSubThoroughfare();

        return add;
    } catch (IOException e) {
        e.printStackTrace();
        Toast.makeText(this, e.getMessage(), Toast.LENGTH_SHORT).show();
        return null;
    }
}

public boolean isSmsPermissionGranted() {
    return ContextCompat.checkSelfPermission(this, Manifest.permission.READ_SMS) ==
    PackageManager.PERMISSION_GRANTED;
}

/* Request runtime SMS permission*/
private void requestReadAndSendSmsPermission() {

    if (ActivityCompat.shouldShowRequestPermissionRationale(this, Manifest.permission.READ_SMS))
    {

    }

    ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.READ_SMS}, 1);
}

public void contact(View v){

    Intent intent = new Intent(Intent.ACTION_PICK, ContactsContract.Contacts.CONTENT_URI);
    startActivityForResult(intent, 1);

}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode == Activity.RESULT_OK) {
        Uri contactData = data.getData();
        Cursor c = getContentResolver().query(contactData, null, null, null, null);
        if (c.moveToFirst()) {

            String name = c.getString(c.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME));
            String contactId = c.getString(c.getColumnIndex(ContactsContract.Contacts._ID));

```

```

        String hasPhone =
c.getString(c.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER));

        if ( hasPhone.equalsIgnoreCase("1"))
            hasPhone = "true";
        else
            hasPhone = "false" ;

        if (Boolean.parseBoolean(hasPhone))
        {
            Cursor phones =
getContentResolver().query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
null,ContactsContract.CommonDataKinds.Phone.CONTACT_ID +" = "+ contactId,null, null);
            while (phones.moveToNext())
            {
                phoneNumber =
phones.getString(phones.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));
            }
            phones.close();
        }
        tv.setText("Phone : "+phoneNumber);
        Log.d("name", name + " num" + phoneNumber + " ");
    }
    c.close();
}
}

}

```

Appendix D: ACTIVITY_MAIN.XML (ANDROID)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    >
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_marginTop="40dp"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <ImageView
            android:layout_width="180dp"
            android:layout_height="180dp"
            android:layout_marginTop="10dp"
            android:background="@drawable/raspberryoff"
            android:id="@+id/imageView" />

        <android.support.v7.widget.SwitchCompat
            android:id="@+id/switchButton"
            android:layout_width="wrap_content"
            android:checked="false"
            android:layout_below="@+id/imageView"
            android:layout_marginTop="50dp"
            android:layout_marginLeft="20dp"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="Raspberry pi"
            /></LinearLayout>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginTop="40dp"
        android:weightSum="1">

        <ImageView
            android:layout_width="160dp"
            android:layout_height="144dp"
```

```
android:layout_marginLeft="20dp"
android:id="@+id/imageView1"
android:background="@drawable/v3"
android:layout_weight="0.31" />
```

```
<Spinner
    android:id="@+id/spinner1"

    android:layout_width="120dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="45dp"
    android:layout_marginLeft="50dp"
/>
```

```
</LinearLayout>
```

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginTop="40dp"
    android:weightSum="1">
```

```
<ImageView
    android:layout_width="160dp"
    android:layout_height="144dp"
    android:layout_marginLeft="20dp"
    android:id="@+id/contact"
    android:background="@drawable/contct"
    android:layout_weight="0.31" />
```

```
<Button
    android:id="@+id/contct"
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:layout_below="@+id/imageView1"
    android:background="#4f9ac2"
    android:layout_marginTop="50dp"
    android:onClick="contact"
    android:layout_marginLeft="50dp"
```

```
        android:text="Contacts"
        android:textColor="#ffffff" />
```

```
</LinearLayout>
```

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginTop="10dp"
    android:weightSum="1">
    <TextView
        android:layout_width="80dp"
        android:id="@+id/tv1"

        android:layout_marginRight="20dp"
        android:layout_height="wrap_content"
        android:layout_below="@+id/contct"
        android:layout_marginTop="10dp"
        android:textColor="#000000"/>
    <TextView
        android:layout_width="wrap_content"
        android:id="@+id/tv"
        android:layout_marginRight="80dp"
        android:layout_height="wrap_content"
        android:layout_below="@+id/contct"
        android:layout_marginTop="10dp"
        android:textColor="#000000"/>
```

```
</LinearLayout>
```

```
</LinearLayout>
```

Appendix E: BASEACTIVITY.JAVA (ANDROID)

```
public class BaseActivity extends AppCompatActivity {

    public static final int MY_PERMISSIONS_REQUEST_LOCATION = 99;

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

        checkLocationPermission();

    }
    private void checkLocationPermission() {
        if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
            != PackageManager.PERMISSION_GRANTED) {
            if (ActivityCompat.shouldShowRequestPermissionRationale(this,
                Manifest.permission.ACCESS_FINE_LOCATION)) {

                new AlertDialog.Builder(this)
                    .setTitle("Location Permission Needed")
                    .setMessage("This app needs the Location permission, please accept to use location
functionality")
                    .setPositiveButton("OK", new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialogInterface, int i) {
                            //Prompt the user once explanation has been shown
                            ActivityCompat.requestPermissions(BaseActivity.this,
                                new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                                MY_PERMISSIONS_REQUEST_LOCATION);
                        }
                    })
                    .create()
                    .show();

            } else {
                ActivityCompat.requestPermissions(this,
                    new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                    MY_PERMISSIONS_REQUEST_LOCATION);
            }
        }
    }
}
```



```

    }
    }
}

@Override
public void onRequestPermissionsResult(int requestCode,
                                     String permissions[], int[] grantResults) {
    switch (requestCode) {
        case MY_PERMISSIONS_REQUEST_LOCATION: {
            if (grantResults.length > 0
                && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                if (ContextCompat.checkSelfPermission(this,
                    Manifest.permission.ACCESS_FINE_LOCATION)
                    == PackageManager.PERMISSION_GRANTED) {}

            } else {
                Toast.makeText(this, "permission denied", Toast.LENGTH_LONG).show();
            }
            return;
        }
    }
}

public boolean isSmsPermissionGranted() {
    return ContextCompat.checkSelfPermission(this, Manifest.permission.READ_SMS) ==
    PackageManager.PERMISSION_GRANTED;
}

/* Request runtime SMS permission*/
private void requestReadAndSendSmsPermission() {

    if (ActivityCompat.shouldShowRequestPermissionRationale(this, Manifest.permission.READ_SMS))
    {

    }

    ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.READ_SMS}, 1);
}

}

```

Appendix F: GPSTRACKER.JAVA (ANDROID)

```
public class GPSTracker extends Service implements LocationListener {

    private final Context mContext;

    // flag for GPS status
    boolean isGPSEnabled = false;

    // flag for network status
    boolean isNetworkEnabled = false;

    // flag for GPS status
    boolean canGetLocation = false;

    Location location; // location
    double latitude; // latitude
    double longitude; // longitude

    // The minimum distance to change Updates in meters
    private static final long MIN_DISTANCE_CHANGE_FOR_UPDATES = 10; // 10 meters

    // The minimum time between updates in milliseconds
    private static final long MIN_TIME_BW_UPDATES = 1000 * 60 * 1; // 1 minute

    // Declaring a Location Manager
    protected LocationManager locationManager;

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

    public Location getLocation() {
        try {
            locationManager = (LocationManager) mContext
                .getSystemService(LOCATION_SERVICE);

            // getting GPS status
            isGPSEnabled = locationManager
                .isProviderEnabled(LocationManager.GPS_PROVIDER);

            // getting network status
            isNetworkEnabled = locationManager
```

```

        .isProviderEnabled(LocationManager.NETWORK_PROVIDER);

    if (!isGPSEnabled && !isNetworkEnabled) {
        // no network provider is enabled
    } else {
        this.canGetLocation = true;
        // First get location from Network Provider
        if (isNetworkEnabled) {
            locationManager.requestLocationUpdates(
                LocationManager.NETWORK_PROVIDER,
                MIN_TIME_BW_UPDATES,
                MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
            Log.d("Network", "Network");
            if (locationManager != null) {
                location = locationManager
                    .getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
                if (location != null) {
                    latitude = location.getLatitude();
                    longitude = location.getLongitude();
                }
            }
        }
        // if GPS Enabled get lat/long using GPS Services
        if (isGPSEnabled) {
            if (location == null) {
                locationManager.requestLocationUpdates(
                    LocationManager.GPS_PROVIDER,
                    MIN_TIME_BW_UPDATES,
                    MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
                Log.d("GPS Enabled", "GPS Enabled");
                if (locationManager != null) {
                    location = locationManager
                        .getLastKnownLocation(LocationManager.GPS_PROVIDER);
                    if (location != null) {
                        latitude = location.getLatitude();
                        longitude = location.getLongitude();
                    }
                }
            }
        }
    }

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

    return location;
}

```

```

/**
 * Stop using GPS listener
 * Calling this function will stop using GPS in your app
 * */
public void stopUsingGPS(){
    if(locationManager != null){
        locationManager.removeUpdates(GPSTracker.this);
    }
}

/**
 * Function to get latitude
 * */
public double getLatitude(){
    if(location != null){
        latitude = location.getLatitude();
    }

    // return latitude
    return latitude;
}

/**
 * Function to get longitude
 * */
public double getLongitude(){
    if(location != null){
        longitude = location.getLongitude();
    }

    // return longitude
    return longitude;
}

/**
 * Function to check GPS/wifi enabled
 * @return boolean
 * */
public boolean canGetLocation() {
    return this.canGetLocation;
}

/**
 * Function to show settings alert dialog
 * On pressing Settings button will launch Settings Options
 * */
public void showSettingsAlert(){

```

```

AlertDialog.Builder alertDialog = new AlertDialog.Builder(mContext);

// Setting Dialog Title
alertDialog.setTitle("GPS settings");

// Setting Dialog Message
alertDialog.setMessage("GPS is not enabled. Do you want to go to settings menu?");

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

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

// Showing Alert Message
alertDialog.show();
}

```