

**EMBEDDED REAL TIME BLINK DETECTION SYSTEM
WITH HEART RATE SENSING FOR DRIVER FATIGUE
MONITORING**



Author

**UMME SAIRA HABIBA- ID 14221001
SHAFINAZ NADIA AHMED-ID 14221002
PRITHILA ANGKAN-ID 14221012
MD ARMAN NOOR SAHAD-ID 14221026**

**A thesis submitted in partial fulfillment of the requirements for
the degree of Bachelor of Science in Electrical and Electronic
Engineering**

Supervisor

Dr. A. K. M. Abdul Malek Azad

Professor

**Department of Electrical and Electronic Engineering
BRAC University, Dhaka**

DECLARATION

We hereby declare that the thesis titled ‘Embedded real time blink detection system with heart rate sensing for driver fatigue monitoring’, a thesis submitted to the Department of Electrical and Electronic Engineering of BRAC University in partial fulfillment of the Bachelor of Science in Electrical and Electronic Engineering is our own work. The work has not been presented elsewhere for assessment. The materials used from other sources have been acknowledged.

Signature of Supervisor

.....
Dr. A. K. M. Abdul Malek Azad

Signature of Authors

.....
Umme Saira Habiba

.....
Shafinaz Nadia Ahmed

.....
Prithila Angkan

.....
Md Arman Noor Sahad

LIST OF FIGURES

Figure 01: Drowsy driver

Figure 02: Block Diagram of the Overall system

Figure 03: Circuit diagram of Overall system

Figure 04: Grove ear clip heart rate sensor

Figure 05: Arduino UNO

Figure 06: Block diagram of heart beat detection using Arduino UNO

Figure 07: The heart beat sensor implemented using Arduino UNO

Figure 08: MCP 3008

Figure 09: MCP3008 pin configurations

Figure 10: Grove ear clip

Figure 11: Raspberry Pi

Figure 12: Raspberry Pi night vision camera

Figure 13: Visual representation of 68 key facial coordinates

Figure 14: Real Time Visual representation of 68 key facial coordinates

Figure 15: Real Time Visual representation of 6 key coordinates of each eye

Figure 16: Coordinates of the eye

Figure 17: Eye Aspect Ratio Equation

Figure 18: Value of EAR over time

Figure 19: Real time computation of EAR

Figure 20: Real time computation of EAR

Figure 21: Flowchart of the code for heart beat detection using Arduino UNO

Figure 22: Code for the heart rate detection using Arduino UNO

Figure 23: Flowchart of the whole system

Figure 24: Blink sensor and heart rate sensor implemented together

Figure 25: Placement of the night vision camera

Figure 26: Implementation of the system in the car

Figure 27: EAR for different people for eyes open and eyes closed

Figure 28: Blink time for number of frames

Figure 29: Graph for EAR vs Luminous intensity

Figure 30: Graph of Heart rate vs Sample

Figure 31: ISDSN Detection system

Figure 32: Placement of alcohol sensor on steering wheel

Figure 33: Proposed Scenario of BSN

Figure 34: The model for future work

Figure 35: System Model of WBAN

Figure 36: WBAN communication alerting the vehicle

LIST OF TABLES

Table 01: Connections from mcp3008 to specific pins of raspberry pi to use SPI connection

Table 02: Specification table of grove ear clip heart rate sensor

Table 03: Experiment Table for different “range_val”

Table 04: Sample and EAR values for both eyes open and eye closed

Table 05: Blink time for number of frames

Table 06: EAR values at different luminous intensity

Table 07: Heart rate of different people when they are awake at different time of the day

Table 08: Costing of the system

ABBREVIATION

ADC- Analog to Digital Converter
BAC- Blood Alcohol Concentration
BPM- Beats Per Minute
BSN- Body Sensor Network
EAR- Eye Aspect Ratio
EDA- Electrodermal Activity
FARS-Fatality Analysis Reporting System
FGPA- Field Programmable Gate Array
GPIO-General Purpose Input Output
GPS- Global Positioning System
ISDSN- Intelligent Steering Wheel Distributed Sensor Network
LED- Light Emitting Diode
NHTSA- National Highway Traffic Safety Administration
OBU- On Board Unit
PDA- Personal Digital Assistant
RSU- Road Side Unit
SPI- Serial Peripheral Interface
V2I- Vehicle To Infrastructure
V2R- Vehicle To Roadside
V2V-Vehicle To Vehicle
VANET- Vehicular Ad Hoc Networks
WBAN- Wearable Body Area Network
WHO- World Health Organization

CONTENTS

List of figures	iii
List of tables	v
Abbreviation	vi
Acknowledgement	ix
Abstract	x
Chapter 1: Introduction	1
1.1 Motivation and background	1
1.2 Literature review	2
1.3 Objective	3
1.4 Overview of content	3
Chapter 2: Project Overview	5
Chapter 3: Heart beat detection	8
3.1 Heart beat detection using Arduino UNO	8
3.1.1 Components used	8
3.1.2 Implementation	10
3.1.3 Result	11
3.2 Heart beat detection using Raspberry pi	12
3.2.1 Components used	12
3.2.2 Biological Background	14
3.2.3 Implementation	14
Chapter 4: Eye blinking detection using Raspberry Pi	15
4.1 Components used	15
4.2 Implementation	18

Chapter 5: Software	24
5.1 Heart beat detection using Arduino UNO	24
5.2 Heart beat detection and Eye blinking detection using Raspberry Pi	26
Chapter-6 Commissioning the whole system	30
Chapter 7: Field test	33
Chapter 8: Conclusion	41
8.1 Summary	41
8.2 Future Prospects	41
8.2.1 Steering wheel	41
8.2.2 BSN, VANET, WBAN	43
8.2.3 Limitation	48
8.2.4 Costing	49
References	50
Appendix	53

ACKNOWLEDGEMENT

We are thankful to our thesis supervisor Dr. A. K. M. Abdul Malek Azad, Professor, Department of Electrical and Electronic Engineering of BRAC University for his guidance for the completion of our thesis. Regards to Aatur Rahman, Project Engineer, CARC, BRAC University for his support throughout the whole thesis work. We are thankful to EEE department, BRAC University for providing us the necessary equipment for the completion of this project.

ABSTRACT

Road accidents are a common phenomenon in our daily lives. Each year these road accidents led to many deaths, fatal injuries and economic losses all over the world. One of the major reasons of these accidents is the drowsiness of drivers. Thus, it is necessary to develop a method to detect the driver's drowsiness to reduce the accident rates. This paper describes a research and project development to detect the drowsiness of drivers. A device was developed that uses two methods to detect the drowsiness. One method was to use the heart beat sensor to check if the heartbeat lies within the threshold value range. The other method was to detect the eye blinking using image processing to check the eyelid's position. The output of both the methods is given as input to the Raspberry Pi. The output from the device is connected to a buzzer that buzzes if the device detects that the driver is drowsy. The final system was implemented and it quite efficiently monitors and detects the driver's fatigue.

CHAPTER-1

INTRODUCTION

1.1 Motivation and background

The increase in the number road accidents is a matter of concern as it is a threat for mankind. Every day in the news we get to hear about road accidents and the loss it has caused. Even in Bangladesh, road accidents are a common phenomenon. These accidents lead to loss of lives, fatal injuries and economic losses. According to the Daily Star [1], at least 2,297 people died in road accidents in Bangladesh in the first half of 2017. Thus, it can be concluded that a large number of these accidents happen due to the drowsy state of drivers. Therefore, it is very necessary to develop a system that can detect the drowsiness of the drivers and implement it in vehicles.



Figure 01: Drowsy driver

Accidents due to drowsy driving occur mostly in vehicles like trucks and buses that travel at night. According to UCLA Sleep Disorder Centre, drivers who work in shifts like night shifts, have sleep disorders like insomnia, take medicines that make people drowsy like cough syrups or those who drink alcohol are likely to drowse off while driving[3]. From our research, we found many different methods to detect the drowsiness such as monitoring the head position, steering wheel pattern, eyelid movement, heart rate, lane deviation. But in our device we have decided to implement two methods. One is the heartbeat detection and the other is the eyelid position monitoring.

1.2 Literature Review

Micro-sleep is one of the major causes of road accidents that take place. Fatigue and drowsiness can be detected by analyzing the condition of the eye. The eyes can be analyzed using a camera that is set up on the dash board. Yet, analyzing the eyes can be limited by lighting conditions. An IR illuminated camera that takes the advantage of bright pupil when exposed to IR can be used for analyzing under any condition. Discussions about such a device which consists of IR illuminated webcam and other devices that work together to detect drowsiness has been made in paper [3]. The outcome of the device depends mainly on effective and efficient image processing techniques. The paper also talks about several mathematical approaches to analyze the eyes, such as Kalman filtering, the Mean shift algorithm, template based correlation. A laboratory model was developed and tested which consisted of an IR camera. The data from the camera was fed to data acquisition card and then to a micro-controller. The micro-controller then analyzed the image. At first, the face region was separated from the whole image using Boundaries Function. Then Region Prop Technique was used to identify the eyes from the rest of the eyes. If the eyes are closed for a significant amount of time, the buzzer goes off and a led in the system turns red. The experiment gave positive results even when different persons were used. It also gave a robust performance at different environmental condition. The writers concluded that it can be used as a very effective device to detect driver drowsiness, thus preventing micro-sleep which leads to accidents.

Different methods like Steering Pattern Monitoring, Vehicle Position in Lane Monitoring, Driver Eye/Face Monitoring, and Physiological Measurement have been suggested in paper [4] that can be used to detect the driver's drowsiness. This paper focuses on a model of Smart Band which monitors the heart rate to detect the drowsiness. The grove ear clip which contains the sensor can be placed on the tip of the finger or the ear lobe. The sensor has light emitting diode and a light detecting receiver like photodiode or light detecting resistor. The heart beat pulse cause the blood flow to vary in different regions of the body. When the grove ear clip is worn the tissues are illuminated by light and depending on the blood volume in that tissue light is absorbed, transmitted or reflected. The reflected light is received by the detector and the detector output is proportional to the heart rate. This output is given as input to the Arduino UNO. The Arduino compares this result with the threshold heart rate in the code given in to it. If the input heart rate is greater than the threshold value the buzzer is off. But if

it falls below the threshold value, it means maybe the driver is drowsy or has fallen asleep and the buzzer is triggered and it rings to wake up the driver. The code in paper [4] is developed in embedded C' language.

1.3 Objective

The main objective of this project is to develop a device that can detect the driver's drowsiness. In order to detect the drowsiness we decided to use two methods. One method is to use the heart beat sensor and check if the heart beat is within the threshold value range. The other method is to detect the eye blinking using image processing. The output from the two methods is given as input to the Raspberry Pi that uses Python language and an overall system is developed. If the heart beat is outside the threshold value range and if the eyelids are found closed then the buzzer would be ON.

1.4 Overview of contents

The rest of the dissertation is organized as follows:

Chapter-2: Project overview

In this chapter, the project overview has been discussed. How we want to develop the system, what we want to implement and what results we expect to get has been discussed in short.

Chapter-3: Heart beat detection

In this chapter, the implementation of the heart beat sensor has been explained in details. This chapter is also based on the discussion of heart beat sensing using Arduino UNO and Raspberry Pi has been discussed. In this chapter the reason for switching from Arduino UNO to Raspberry Pi has been stated.

Chapter 4: Eye blinking detection using Raspberry Pi

This chapter contains discussions about eye blinking detection in details. The way the image is obtained, image processing is done and how results are obtained has been discussed.

Chapter 5: Software

In this chapter, the flowcharts of the codes and the portion of our contribution to the code has been explained.

Chapter 6: Combining the overall system

In this chapter, the methods of how the system has been combined together has been explained in details.

Chapter 7: Field test

The final system has been tested in different places at different condition. The details of the tests and the results obtained have been discussed in this chapter.

Chapter 8: Conclusion

In this last chapter the main results of the system has been summarized and some concluding remarks and some directions for future works has been provided.

CHAPTER-2

PROJECT OVERVIEW

In this chapter, the overview of the system has been discussed in details. This chapter contains brief explanation of the hardware that we used to build up the system. This chapter also contains the discussions of the software and the processor that has been used.

From the literature review, we found out that there are many ways to detect the driver's drowsiness. In our system, we have decided to implement two methods to detect the drowsiness. One is to monitor the heart beat and the other is to monitor the eye blinking. Any one of the two methods can effectively predict the driver's drowsiness but to make the system more efficient we have incorporated both the methods together.

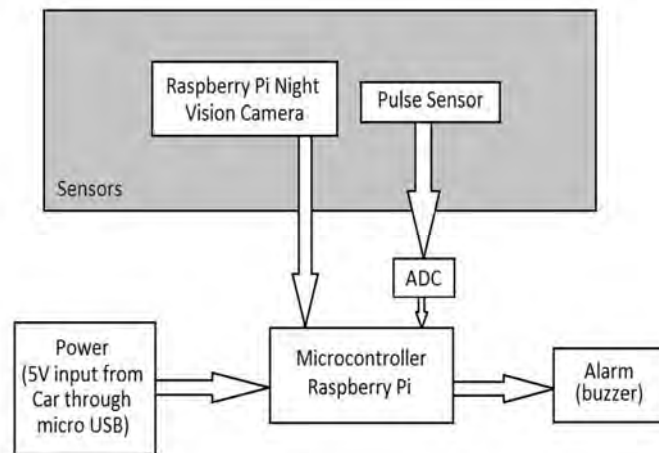


Figure 02: Block Diagram of the Overall system

The block diagram shows the design of the overall system. The Raspberry Pi night vision camera and the pulse sensor are the two sensors we will implement in the system. The signal from the Raspberry Pi night vision camera directly goes to the Raspberry Pi but the signal from the pulse sensor passes via the ADC. The Raspberry Pi is powered from the car by USB cable. The output from Raspberry Pi goes to the buzzer.

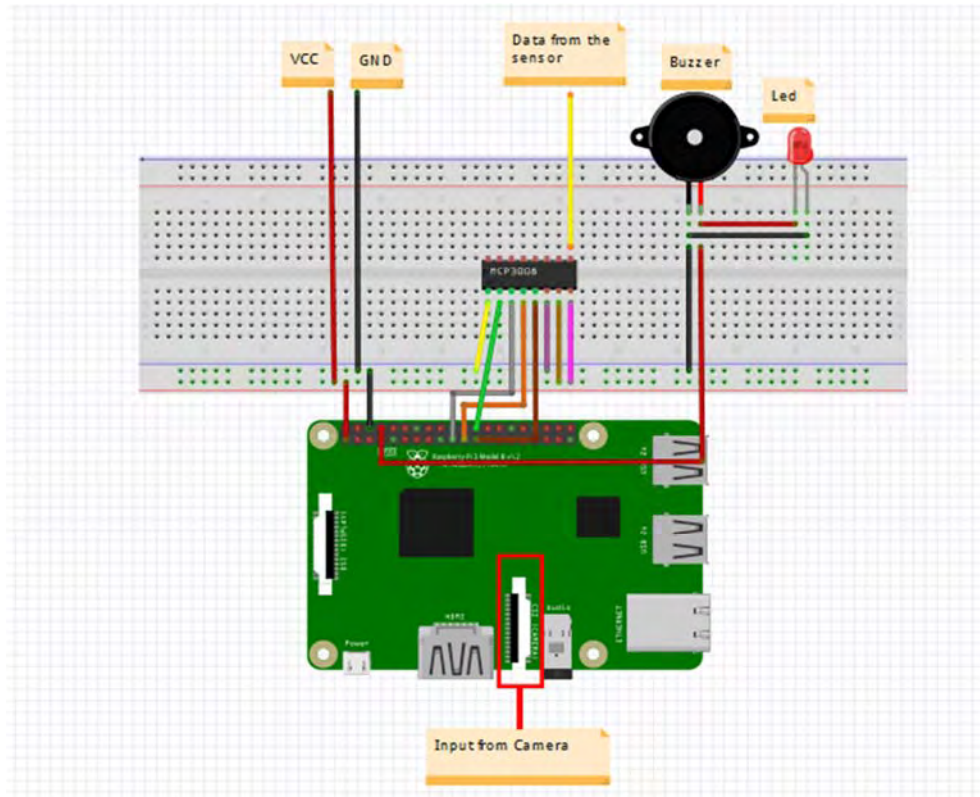


Figure 03: Circuit diagram of Overall system

The diagram shows the design of our overall system. It shows all the hardware that we needed to build up the system. It consists of a pulse sensor, Raspberry Pi Night Vision Camera, ADC, USB cable, Raspberry Pi, LED and alarm.

The pulse sensor measures the heart beat and Raspberry Pi Night Vision Camera for eye blinking detection and these are the two sensors of our system. These two sensors provide the input signals. The pulse sensor has a grove ear clip that can be worn on the fingertip or the ear lobe according to the convenience of the driver. For conversion of the signal received from the pulse sensor an ADC is incorporated in the system. The pulse sensor measures the heart beat and sends the signal to the Raspberry Pi via the ADC. A Raspberry Pi night vision camera is placed on the dashboard. The camera detects the face and the facial features. A facial landmarks detector is used to find the facial coordinates. The night vision camera detects if the eyes of the driver are open or closed.

We first used the Arduino UNO to implement the heart beat sensor but due to certain limitations we had to shift to Raspberry Pi. The processing ability of Raspberry Pi is more than that of Arduino UNO which makes it compatible for image processing. Hence, for the whole system Raspberry Pi was used as the processor. Raspberry Pi needs to power to operate. USB cable is required to power up the whole system and this power is provided from the car.

For the software, we combined the codes for the eye blink detection and heart beat sensing. The code for eye blink detection calculates the EAR value. The code for heart beat sensing detects if the input heart beat is less than or greater than the threshold value range.

In the Raspberry Pi code, the Eye Aspect Ratio is calculated. The EAR for each eye is calculated using 6 of the eye coordinates and the EAR calculation formula and the mean value is taken. A threshold value of the EAR is set in the code. If the EAR value is less than the threshold value then it detects the EAR for 5 more frames. If the EAR is still less than the threshold than it will send a HIGH input to the Raspberry Pi. But if in between calculating the EAR for the other 5 frames, the EAR increases then the code will calculate the EAR from the start for 6 frames.

The code has been developed in such a way that only if it detects that the EAR is less than the threshold then it calculates the heartbeat. In the Raspberry Pi code, a minimum and maximum threshold heart beat value is set for heart beat detection. If the input signal falls outside the threshold value range it sends a HIGH signal to the output otherwise a LOW signal goes to the output.

If both the output signals are high then the buzzer goes ON to wake up the driver. It can be concluded that the driver is drowsy or has fallen asleep. If both or even one signal is LOW then buzzer is OFF. With the buzzer we have also connected an LED that glows if both the sensors give a HIGH output.

In this chapter, the details of what we plan to, how we plan to do it and what results we expect have been explained. Discussions about what sensors we will use in the system, the hardware, software and processor we will use have been provided.

CHAPTER-3

HEART BEAT DETECTION

Introduction

In our system we decided to implement two methods. Among them one of the methods that is the heart rate detection was first implemented using the Arduino UNO. In this chapter, the heart rate detection using Arduino UNO has been discussed in details. But due to certain limitations the Arduino UNO was not suitable for our system so we had to switch to Raspberry Pi. The heart rate detection using Raspberry Pi has also been discussed in this chapter.

3.1 Heart beat detection using Arduino UNO

3.1.1 Components used

Grove ear clip heart rate sensor



Figure 04: Grove ear clip heart rate sensor

Grove ear clip heart rate sensor has ear clip and receiver module. The ear clip has heart rate sensor in it and it can be worn on the ear lobe or at the tip of the finger according to the convenience of the user. It has low power consumption, high sensitivity and it is very

convenient to use. Grove ear clip heart rate sensor is connected to Analog to Digital Convertor (ADC) that has three wires:

- Black wire- Ground
- Red wire- Vcc
- Yellow wire- For data transfer to Arduino

Arduino UNO



Figure 05: Arduino UNO

The Arduino Uno is a microcontroller board which is based on the ATmega328. It has total 14 digital pins which can be used as input or output. Among them 6 can be used as PWM outputs. There are also 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button[5]. It is easy to use. Arduino UNO has a USB cable that can be connected to the laptop/computer to get the power or it can be easily powered using a battery.

3.1.2 Implementation

The block diagram below shows how to set up the circuit.

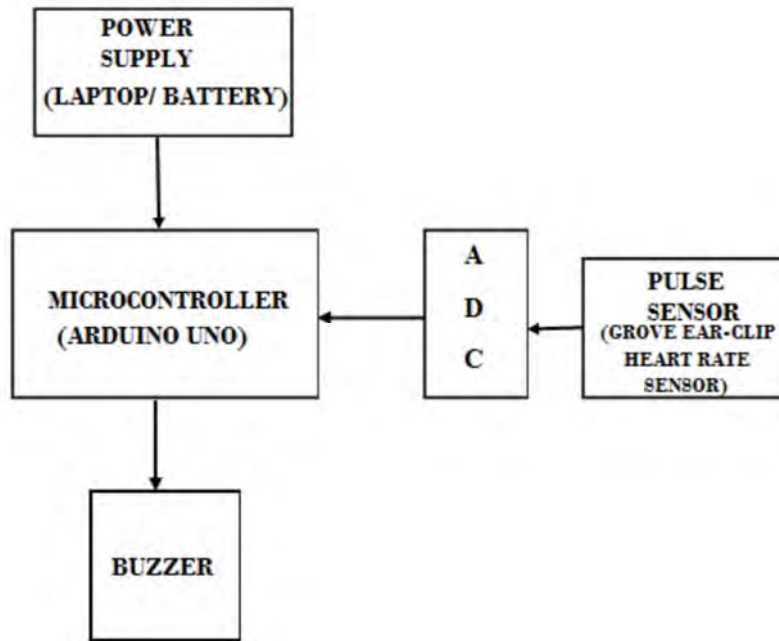


Figure 06: Block diagram of heart beat detection using Arduino UNO

Arduino Uno and Grove Ear-Clip heart rate sensor were used for making the heart beat sensor circuit. The Grove Ear-clip heart sensor can be worn by the driver on the fingertip or the ear lobe according to the convenience of the driver. The sensor emits Infra-Red light at the earlobe or figure tip and the light is reflected off the tissue. The reflected light is detected by the detector in the sensor and the heart rate is measured.

The Grove Ear-clip heart sensor is attached to the Analog to Digital Convertor, which has three wires attached to it. We connected one wire (yellow) to digital pin number 2 for input from the sensor, one to the power (red) and one to the ground(black)of the Arduino. The Arduino drew 5V power from the computer it was connected to. The buzzer was connected to pin number 11 and ground of the Arduino Uno.

After taking 20 inputs from the heart rate sensor the Arduino calculates the average heart rate. A threshold heart rate is set in the code. The threshold was set to 60 in the code. If the calculated heart rate is less than the threshold heart rate the buzzer is on and the driver is alert. Otherwise, the buzzer remains off.

3.1.3 Result

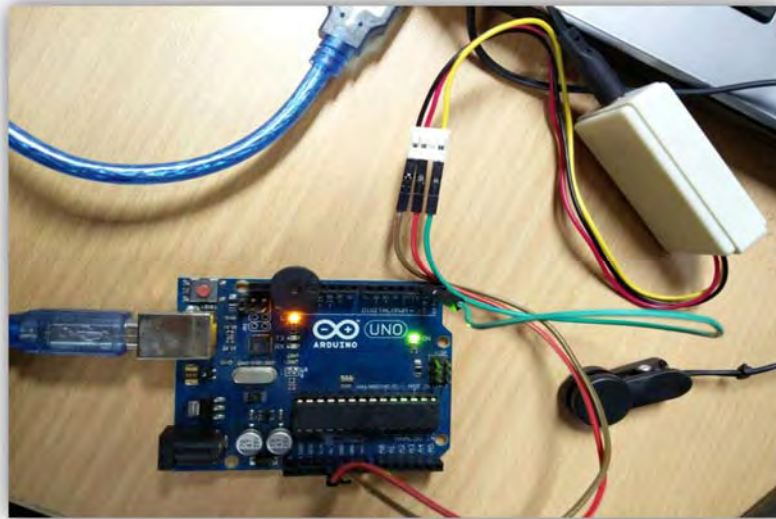


Figure 07: The heart beat sensor implemented using Arduino UNO

The heart beat sensor was implemented by connecting the hardware as shown in the above figure. The system accurately measures the heart beat and it was quite efficient.

3.1.4 Limitation

The sensor accurately measures the heartbeat. The processor used for this is the Arduino UNO. But the Arduino is not compatible for image processing. In our system we want to implement two methods and for the second method we want to incorporate in our project we need image processing but for that we cannot use Arduino. That is why we had to change the processor from Arduino to Raspberry pi.

3.2 Heart beat detection using Raspberry pi

3.2.1 Components used

MCP 3008

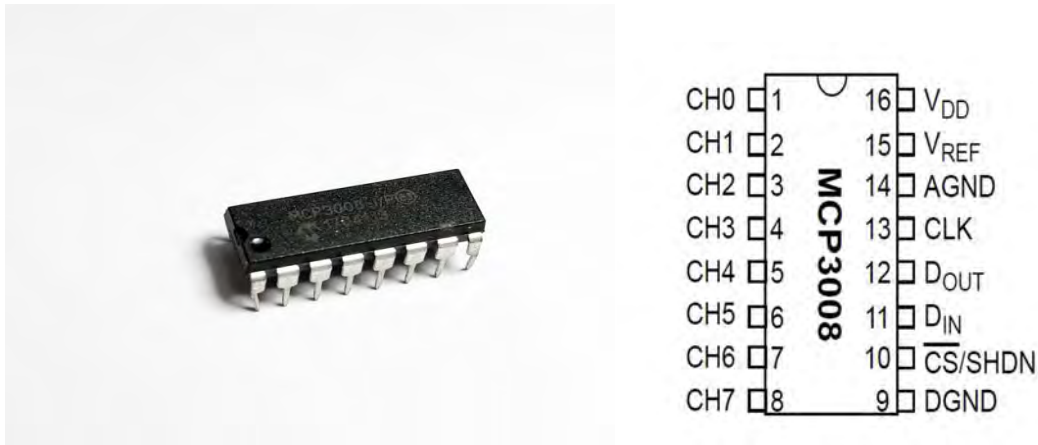


Figure 08: MCP 3008 Figure 09: MCP3008 pin configurations

- MCP 3008 is an ADC (analog to digital converter)
- It reads the raw data transferred and collected from the sensor and converts it to digital data [6].
- GPIO pin and SPI software is used to connect to the ADC. Another option is to use the SPI hardware. But to minimize the cost we used the software instead. Also the hardware has some limitations and could work with certain pins only [6].
- 8 channels to read the analog signals and send it to the raspberry pi [6].
- The digital data is 10 bit [6].
- SPI (serial peripheral interface) serial connection required for connection
SPI is used to send data between microcontrollers and devices such as sensors, registers and SD cards. To select the required device it has a separate clock and select line [7].

The table for the specification pins of the ADC is given below [6].

MCP3008	Raspberry Pi
VDD	3.3V
VREF	3.3V
AGND	GND
DGND	GND
CLK	Pin 18
DOUT	Pin 23
DIN	Pin 24
CS/SHDN	Pin 25

Table 01: Connections from mcp3008 to specific pins of raspberry pi to use SPI connection

Grove ear clip heart rate sensor



Fig 10: Grove ear clip

- Consist of an ear clip and a receiver module
- Low power consumption [8].
- High sensitivity [8].
- Three wires for ground, Vcc and data

The specification table of the grove ear clip heart rate sensor is given below, see [8].

	minimum	typical	Maximum	Unit
Voltage	3.0	5.0	5.25	V
Work current	6.5			mA
Length of ear clip wire	120			cm

Table 02: Specification table of grove ear clip heart rate sensor

3.2.2 Biological Background:

The normal heart rate also known as pulse varies from person to person. Generally, for adults the normal heart rate is within the range of 60 to 100 beats per minute (bpm)[9]. During exercise the maximum heart rate of an adult could be up to 200 bpm[10]. On the other hand heart rate could be elevated because of fatigue, stress, anger or other emotional turmoil as our heart tries to pump oxygen to increase our energy [11]. While sleeping the normal heart rate of adults would drop till 24bpm, however, for people of around 80 years old the sleeping heart rate could be 14bpm[12].

3.2.3 Implementation

The grove ear clip can be worn on the ear lobe or the fingertip according to the preference of the driver. Raw data is collected via the heart rate sensor. An ADC is connected to the grove ear clip that data from analog to digital signal. This signal is given as input to the Raspberry Pi as input.

A threshold minimum heartbeat is set on the code. The heart beat from the sensor is compared with the threshold heartbeat. If the heartbeat recedes below the minimum heartbeat rate the buzzer will be turned on showing that the driver maybe drowsy. The condition of the driver is considered as normal if the heartbeat is above the threshold heart beat and the buzzer will be off.

Conclusion

In this chapter, the heart rate detection using both the Arduino UNO and Raspberry Pi has been explained in details. Also the reason for switching from Arduino to Raspberry Pi has been discussed in this chapter.

CHAPTER-4

EYE BLINKING DETECTION USING RASPBERRY PI

Introduction

In our system, we decided to implement two methods. We have already discussed the first method. In this chapter, the details for the second method that is the eye blinking detection have been discussed. The components used, the methods used and the results have been discussed.

4.1 Components used

Raspberry Pi



Top view



Side view

Figure 11: Raspberry Pi

The microcontroller used for the project is a Raspberry Pi 3 Model B, which is a credit card sized Single Board Computer. The Raspberry Pi 3 Model B is a third generation model and is 50% faster than its previous model [13]. It has a Quad Core 1.2GHz Broadcom BCM2837 64bit CPU, 1 GB of RAM, 400 MHz VideoCore IV® GPU[14]. It supports 802.11n wireless LAN and Bluetooth 4.1 connectivity. Unlike other

microcontrollers, there is no need to connect any external antenna. The antenna is soldered on the board to keep the size of the device small. Even though the antenna is small, it is capable enough to pick up wireless signals even through the walls [14]. There are 4 usb 2.0 ports, 4 Pole stereo output and composite video port, Full size HDMI, CSI camera port for connecting a Raspberry Pi camera, DSI display port for connecting a Raspberry Pi touchscreen display. HDMI cable is used to connect the microcontroller to a HDMI supported display, and then the microcontroller can be programmed using keyboard. The use of HDMI can be omitted when through remote desktop connection, when both the controller and the computer are in the same network. Mice, keyboards, network adapters and external storage can be connected using the USB ports.

There are 40 extended GPIO pins and they provide 4 different function-GPIO, Ground and Power [15]. There are 2 different supply pins in Raspberry Pi 3 Model B. The 3V3 supply pin can provide up to 500mA, but using the 5V supply is preferable. There are 8 available ground pins and they are all electrically connected.

There are no ON/OFF switches available to turn on the device. Supplying power to it will automatically boot it up[16]. It can be turned off by disconnecting the power supply, or when in the graphical environment, it can be shut down from the main menu or using the command terminal to execute code that will turn it off. The microcontroller uses a 5V USB input, also has ports where external power source can be connected to provide power.

The microcontroller used in this project uses operating system Raspbian, with the version Jessie. It is a lightweight and is Debian-based Linux operating system. It is specifically designed to work for Raspberry Pi devices [17]. A lightweight LXDE desktop allows a user friendly interface to work with and uses python as the coding language. The OS is installed in an 8GB memory card and this allows us to install additional packages and make programs of our own.

Raspberry Pi night vision camera



Fig 12: Raspberry Pi night vision camera

The camera used in the project was a Raspberry Pi Night Vision Camera. The camera is connected to the microcontroller through the CSI connector. It has two high intensity infrared LED spotlights which obtains power directly from the CSI port. They are capable of lighting an area up to a distance of 8 meters [18]. There is a tiny adjustable potentiometer on each LED board that can be used to change the threshold. The camera specifications are given below [18]

- 5 Megapixel OV5647 Camera
- 2 x 3W high-power 850 infrared LEDs:
- Onboard photoresistor to detect ambient light
- Onboard adjustable resistor, for controlling the ambient light threshold of toggling the infrared LED
- CCD size : 1/4inch
- Aperture (F) : 1.8
- Focal Length : 3.6MM (adjustable)
- Diagonal : 75.7 degree
- Sensor Resolution : 1080p

4.2 Implementation

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 centre, the mouth corners and centers. Our system uses a facial landmark detector that is included in the dlib library. This detector is an implementation of *One Millisecond Face Alignment with an Ensemble of Regression Trees* paper by Kazemi and Sullivan (2014). Results are achieved faster and effectively with the help of 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 the diagram below.

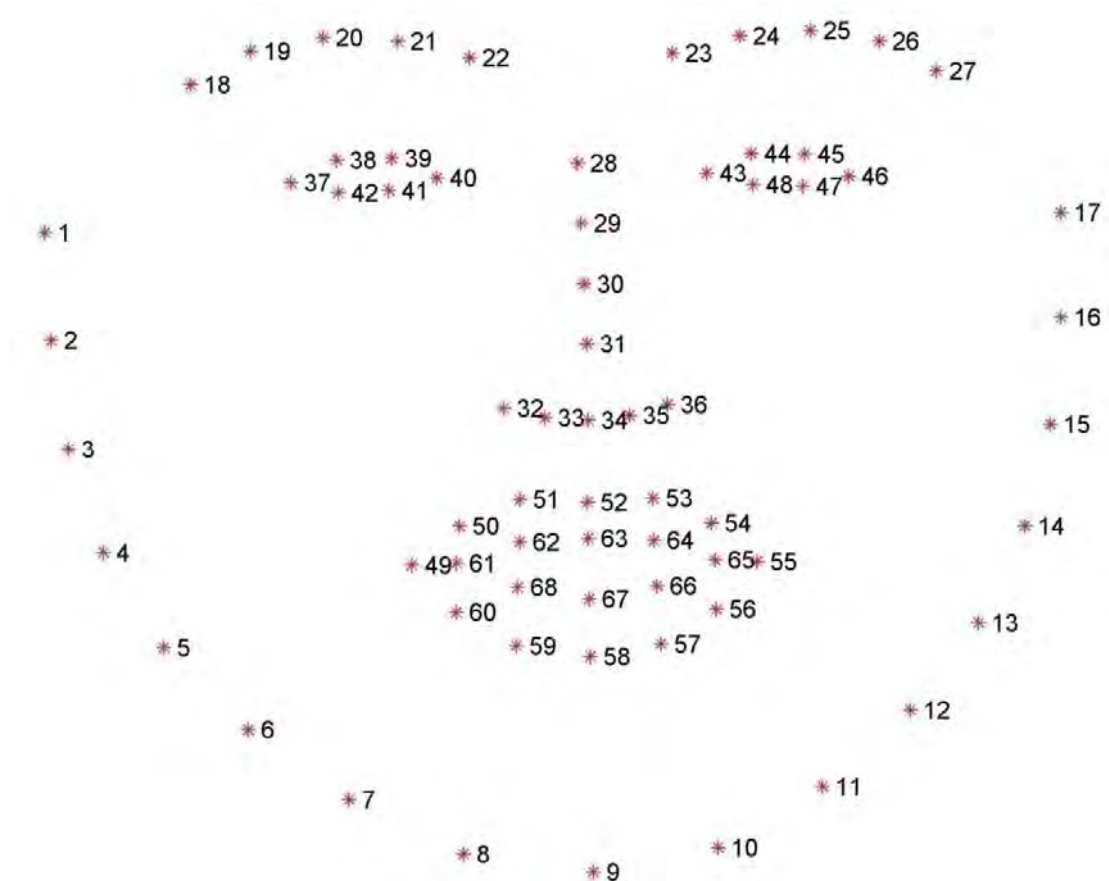


Fig 13: Visual representation of 68 key facial coordinate

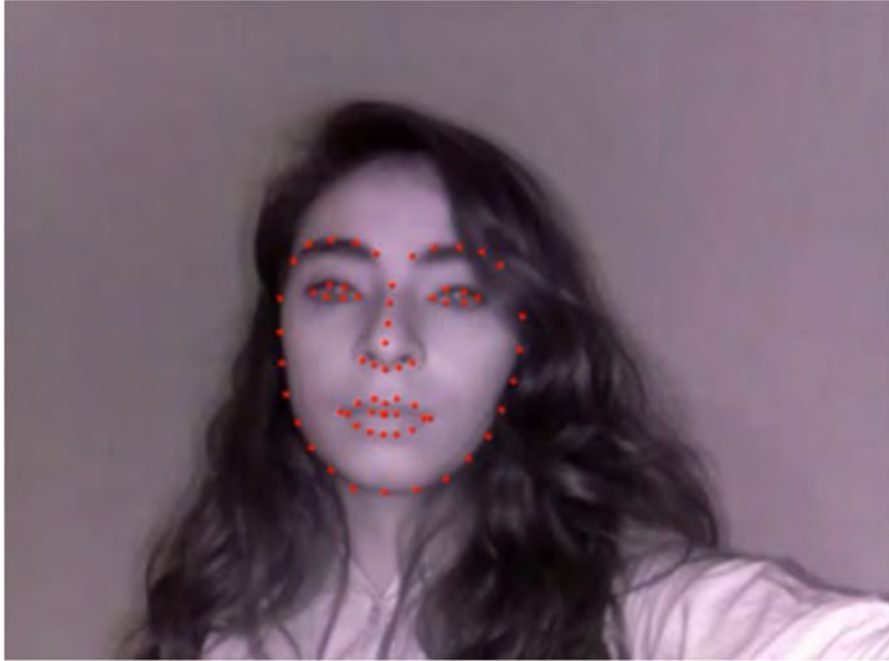


Figure 14: Real Time Visual representation of 68 key facial coordinates

The coordinates of both the eyes are very important to us because that will help us to find out if the eyes are open or not. Each eye has 6 unique coordinates, 2 on the top of the eye, 2 on the bottom and 1 on each side horizontal edge of the eye.

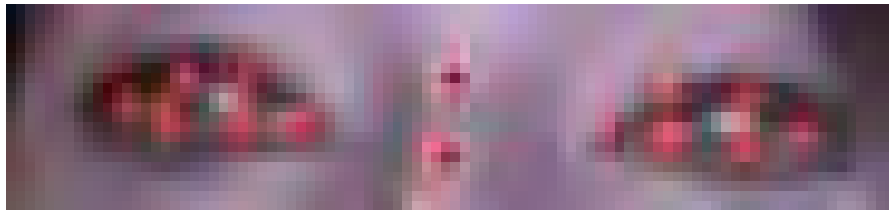


Figure 15: Real Time Visual representation of 6 key coordinates of each eye

These 6 particular coordinates are used to find the Eye Aspect Ratio, EAR. The EAR is a formula that is discussed in *Real-Time Eye Blink Detection using Facial Landmarks* by Soukupova' and Čech (2016). Their paper[19] proposed a formula that detects eye blink using the scalar quantity EAR. 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. At the top-left in fig 5, the eye is fully open and the landmarks are visible along with the 6

key points, and the plot below shows that the EAR is constant over time. At the top-right in Fig 5, the eye is closed and thus the EAR drops significantly.

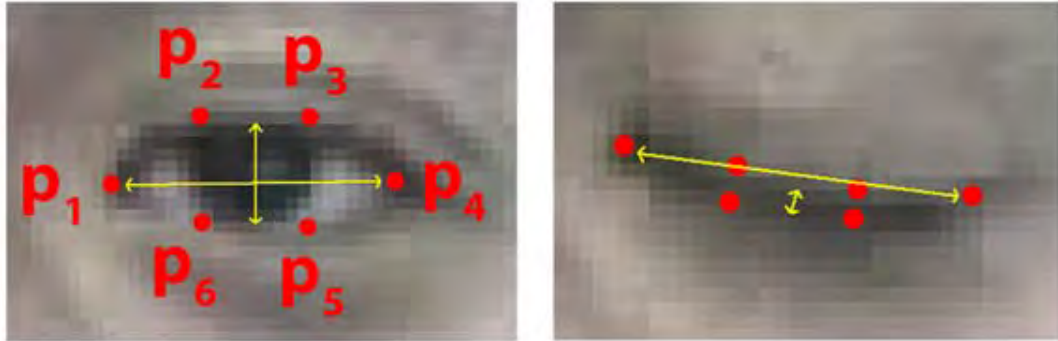


Figure 16: Coordinates of the eye

Source: Adapted from [19]

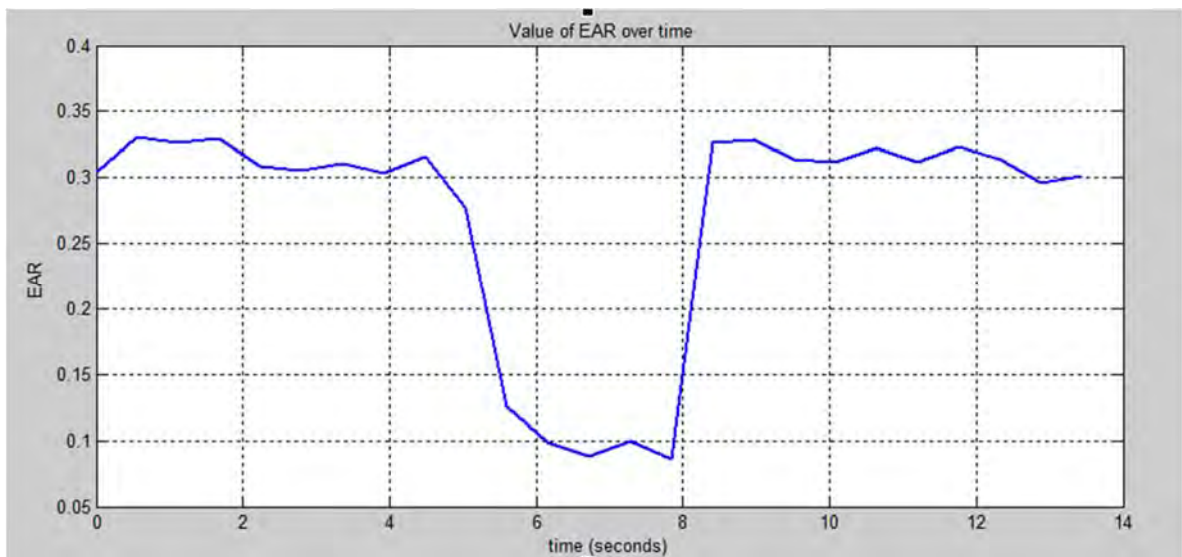


Figure 17: Value of EAR over time

Each 6 coordinates are named as P1 to P6 starting from the left edge of the eye and going clockwise with the remaining coordinates.

$$\text{EAR} = \frac{\|p2 - p6\| + \|p3 - p5\|}{2\|p1 - p4\|}$$

Figure 18: Eye Aspect Ratio Equation

Source: Adapted from [19]

The distance between two points, such as $p2$ and $p6$ 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 6 computes the distance between vertical coordinates and the numerator calculates the distance between the horizontal coordinates. This formula saves us time to find out if the eyes are closed, because conventional method for checking if they are closed relies on further image processing.

Use of EAR is non-intrusive and allows us to find out the amount to which the eyes are open, and determine whether they are open or closed. Other methods to find if the eyes are closed require more processing power. One of the methods includes the use of illuminated pupil. In *Real Time Eye Detection and Tracking Method for Driver Assistance System* [3] by Ghosh, Nandy and Manna (2015) the eyes are exposed to IR rays and they reflect it producing a bright pupil effect. This particular method says that the eyes are open if bright pupil is detected, or the eyes are closed if there are no bright pupils. More work has to be carried out by the processor to find out for bright pupil, which can cause the system to run slower. But EAR relies on simple mathematics and results are fast and accurate.

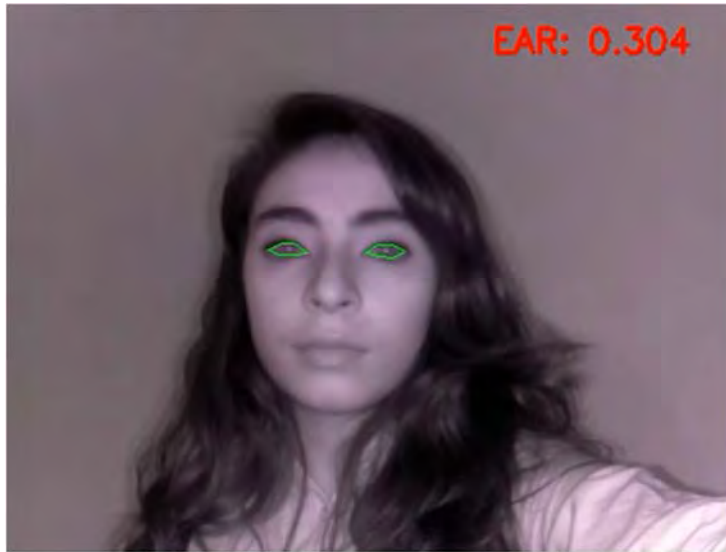


Figure 19: Real time computation of EAR

The EAR value for both the eyes is calculated and then the mean is taken. The EAR is calculated for each consecutive frame and a threshold for EAR is set in the code. If the value of EAR is less than the threshold, a counter is increased. If the value of the counter reaches 6, it can be deduced that the eyes are closed, or not open enough. If that happens, an alert is passed onto display output, along with the GPIO pins that trigger an alarm to alert the driver.



Figure 20: Real time computation of EAR

Conclusion

The components and methods used for eye blinking detection have been discussed in this chapter. The EAR value calculation and how it can be used to detect the driver's fatigue has also been discussed in this chapter.

CHAPTER-5

SOFTWARE

Introduction

The codes used for the heart rate detection using both Raspberry Pi and Arduino UNO and also for eye blinking detection have been explained in this chapter.

5.1 Heart beat detection using Arduino UNO

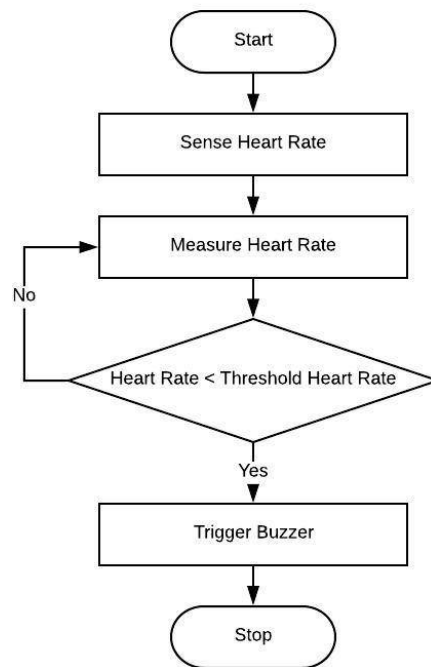


Figure 21: Flowchart of the code for heart beat detection using Arduino UNO

The grove ear clip worn on the ear lobe or the fingertips takes the heart beat as input signal. Then the input is converted to digital signal from analog signal using an ADC. The signal from the ADC goes to the Arduino UNO. In the Arduino the checks if the heart beat is greater than or less than the threshold value. If the heart rate is less than the threshold value then the buzzer goes ON. But if the heart rate is less than the threshold value then the buzzer is OFF.

```
if(heart_rate < 60)
{
  Serial.println("Buzzer ON");
  digitalWrite(LED, HIGH);
  delay(1000);
}
else
{
  Serial.println("Buzzer OFF");
  digitalWrite(LED, LOW);
}
```

Figure 22: Code for the heart rate detection using Arduino UNO

Based on the flowchart the code has been developed. The above portion of the code shows that we have set the threshold heart beat value to 60. This value '60' was found out by doing a lot of research. The particular portion shows how the Arduino determines to turn ON or keep the buzzer OFF the buzzer based on the input heart beat and the threshold value.

5.2 Heart beat detection and Eye blinking detection using Raspberry Pi

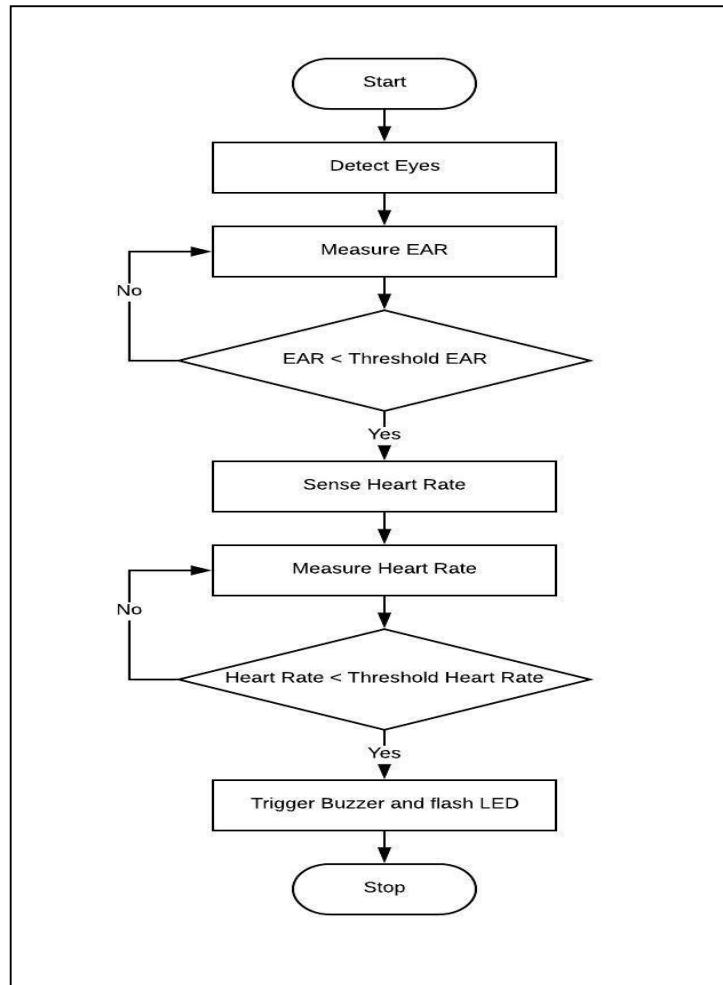


Figure 23: Flowchart of the whole system

The flowchart shows that first the system detects the eyes using the night vision camera. The source code detecting the eyes is taken from pymimagesearch.com [20]. The EAR is calculated. If the EAR is greater than the threshold it measures the EAR again. But if EAR is less than the threshold than it measures the heartbeat. If the heart beat is less than the threshold than it triggers the buzzer and flashes the LED.

```

#This parameters filter noise value by accepting fixed number of BPM values
#which exists within a valid range

last_bpm = 0 #The Last BPM read from the Sensor class
bpm_in_range = 0 #Number of BPM value detected which is within the range
bpm_max_val = 3 #Number of consecutive BPM values that must exist within a valid range

#Maximum difference between last value read and current value read
range_val = 20 #Maximum difference between last and current BPM value that validates range
processing = False #The detected BPM value was obtained while calculating BPM or from the sensor

```

The source code for measuring heart beat is retrieved from tutorials-raspberrypi.com [21]. But there was only one threshold given whereas we needed two threshold values. So we modified the code such that a minimum and a maximum thresholds are set with which the heartbeat of the driver is compared. If the heartbeat of the driver remains within the threshold range the buzzer will not be turned on. In the grove sensor we are using noise is not filtered automatically. As a result any external vibration or sound effect the values of the sensor. The sensor sometimes gives BPM value over the normal heartbeat range while the person is in normal physical state [22]. Also sometimes the sensor gives a high BPM output even when the sensor is not connected to the fingertip or the earlobe [23]. To get rid of those external values we needed to build a low pass filter. The number of consecutive pulses that would be taken for comparison is denoted as “bpm_maxval”. Here we are comparing the previous 3 pulses with the current pulse. The “range_val” is the difference between the driver’s current pulse with the previously obtained pulses. Here it is set as 20. That means if the differences among the previous 3 pulses are within 20 with the current pulse then only the current pulsed will be taken in as heartbeat. Otherwise the value will be discarded. While experimenting we found that the noise that is interfering with the sensor’s value is very above 50. And the general fluctuation between consecutive 3 heartbeats of a human is less than 20. Therefore, for eliminating the noise the difference between the 3 consecutive heartbeats is set as 20.

```

# draw an alarm on the frame
cv2.putText(frame, "DROWSINESS ALERT!", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
#GPIO.output(24, 1)
print "-----EYES CLOSED!!!!!!!!!!!!!!!!!!!!!!-----"
#-----DETECTION OF HEARTBEAT PULS-----
#buzzer_off()
light_on()
bpm = p.BPM
if bpm == 0:
    processing = False
else:
    processing = True
print "Difference: " + str(fabs(bpm - last_bpm))
if fabs(bpm-last_bpm) <= range_val:
    if bpm_in_range >= bpm_max_val:
        bpm_in_range = bpm_max_val - 1
    else:
        bpm_in_range = bpm_in_range + 1
        bpm = 0
elif bpm != 0:
    last_bpm = bpm
    bpm_in_range = 0
    bpm = 0

```

As soon as the drowsiness alert is detected the code will go for checking the heartbeat. The heartbeat will only be accepted if the differences among the previous three consecutive values are within 20 BPM. After that it will check if the heartbeat obtained is within the threshold value or not. If not only then the buzzer will be turned on.

```

p.BPM = 0
print "Last BPM: " + str(last_bpm)
print "No. of BPM in range: " + str(bpm_in_range)
if bpm > 0:
    print("-----BPM: %d-----" % bpm)
    if bpm < min_bpm or bpm > max_bpm:
        buzzer_on()
else:
    if processing:
        print("-----Calculating BPM-----")
    else:
        print("-----No Heartbeat found-----")
    processing = False

```

While comparing with the threshold values and noise filtering the code will show the message “Calculating BPM”. If the sensor gets disconnected from the body or is unable to take input of the heartbeat then it will show the message “No heartbeat found”.

Conclusion

This chapter shows the code for heart beat detection using Raspberry Pi and Arduino and eye blink detection using Raspberry Pi. The important portions of the codes have also been given here and explained in details.

CHAPTER-6

COMMISSIONING THE WHOLE SYSTEM

In this chapter, it has been discussed about how to implement the whole system in the car. This chapter contains details about how to place the camera, the grove ear clip and Raspberry Pi. This chapter basically explains how the hardware and software are combined together.

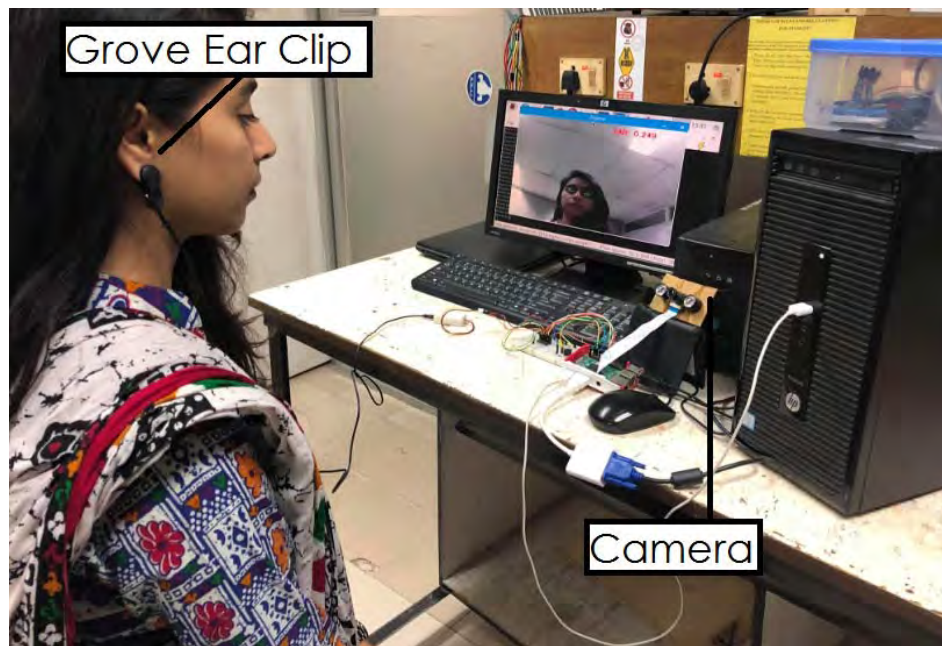


Figure 24: Blink sensor and heart rate sensor implemented together

Both the night vision camera and the grove ear clip are connected to the Raspberry Pi. The same Raspberry Pi is used to process the signals given as input by the two sensors. The Raspberry Pi runs the code. The night vision camera and the Raspberry Pi placed on the dashboard as shown in the above figure. The grove ear clip is worn on the ear lobe. The whole system is powered by supply from the car. The night vision camera is placed in such a way that it does not cause any hindrance to the driver while driving. The night vision camera detects the blinking of the eyes. It is the first sensor input. It takes images of the eyes and using the code the EAR is calculated. If the EAR is less than the threshold value set in the code then it calculates the EAR for 5 more frames. If the EAR for 6 consecutive frames is less than the threshold only then it measures the heartbeat.



Figure 25: Placement of the night vision camera

The camera needs to be adjusted according to the position of the driver's eyes. So, a tripod has been used to place the night vision camera in the dashboard. Using the tripod, the height and position of the camera can be adjusted according to the needs of the drivers.

The grove ear clip contains the heart rate sensor. The driver wears it on the ear lobe during driving. Once the night vision camera gives a drowsiness alert and shows the eyes are closed only then the heart beat is measured. The heart beat measured by the heart rate sensor is the second input. For the heart beat sensor, the code does not measure filter the outside noise so a filter is included in the code. In the code, minimum and maximum heart rates are given as threshold values. If the heart rate is outside the threshold value range then it triggers the buzzer and the buzzer goes ON to show that the driver is drowsy. An LED is also connected at the output that glows alongside the buzzer if the EAR and heart beat are less than the threshold.

The Raspberry Pi has been used as the processor for this system. The Raspberry Pi requires 5V supply so it has been powered using the supply from the car. The Raspberry Pi is placed in box that has been placed with the night vision camera and the grove ear clip connected to it.

Conclusion

This chapter shows how the hardware components were placed in the car. It explains about how the code is processed in the Raspberry Pi to give the output. This chapter is basically the details of how the hardware components and the software are combined to get the overall system.

CHAPTER-7

FIELD TEST

Introduction

In this chapter, we have included all the results that we have obtained during the field test. We have done 4 field tests. This chapter contains all the results.



Figure 26: Implementation of the system in the car

1.Experiment Table for different “range_val”:

Range_val	BPM (beats per minute) 10 values	Description
0	No values	Its very rare that the consecutive BPM are exactly the same. Generally there is a small change in the consecutive BPM.
2	X X X	Very few values could be detected where the difference between the consecutive beats are 2 or less.

	94 X 91 91 X X	
10	93 95 92 98 89 X 90 89 87 91	Almost all the results are reliable when the person is in normal condition. However, while talking the 6 th value the person coughed and the heartbeat jumped up beyond the acceptable difference of BPM. Therefore the value was discarded.
20	90 88 87 85 89 95 112 90 88 87	All the values are reliable. During the 7 th reading the person coughed and the BPM went up by 17. The code did not discard the value as the acceptable difference is set to 20.
60	88 85 87 90 144 88 88	The 5 th reading here way higher than the normal BPM of that particular person. The difference between the 4 th and the 5 th value is 54. However, during the whole time the person was relaxed, did not cough nor sneezed. This is the effect of external noise taken in by the sensor.

	89	
	87	
	90	

Table 03: Experiment Table for different “range_val”

According to the experiment the most suitable value for “range_val” is 20. It eliminates the external noises in addition to that it allows the maximum change in BPM to take as input in a normal human being.

2.Experiment for Sample vs EAR:

The EAR values for 7 different people were recorded. The EAR for both eyes opened and closed was measured. For accuracy the EAR was calculated for 5 times and the mean value was recorded. The table below shows the mean values. Using the values a graph was plotted.

Table for sample vs EAR:

Sample	EAR for eyes open	EAR for eyes closed
1	0.374	0.218
2	0.253	0.121
3	0.350	0.129
4	0.302	0.147
5	0.379	0.195
6	0.275	0.136
7	0.281	0.110

Table 04: Sample and EAR values for both eyes open and eye closed

GRAPH:

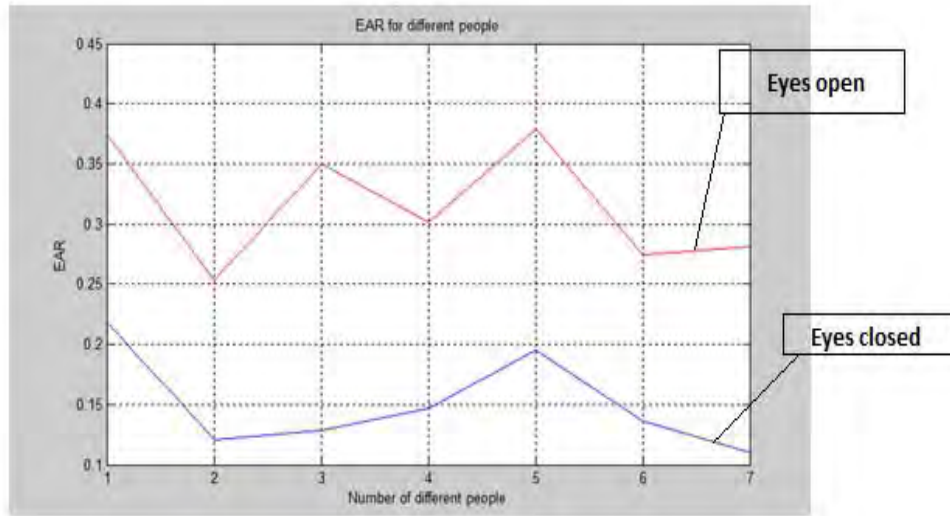


Figure 27: EAR for different people for eyes open and eyes closed

From the graph we can see that, the minimum EAR for eyes open is for Sample2 which is 0.253. The maximum EAR for eyes open is for sample 1 is 0.218. Hence, the threshold was set to 0.235 in the code.

3. Blink time for number of frames

The blink time for different number of frames was measured to find the most appropriate number of frames to set in the code. Using the values a graph was plotted shown below.

Frames	Blink Time(second)
4	1.74
6	2.49
8	3.33
10	3.70
12	4.05
15	5.56
20	6.99
25	9.82

Table 05: Blink time for number of frames

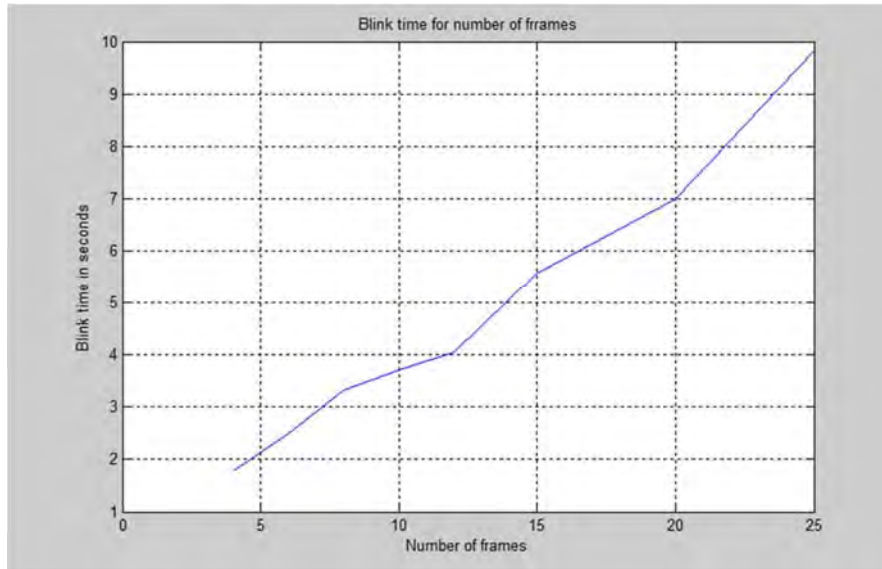


Figure 28: Blink time for number of frames

According to the graph, the most appropriate number of frames is 6. So in the code the number of frames for which the code runs to detect the EAR was set to 6.

4. Experiment to determine EAR is constant at all luminous intensity:

The EAR for the same person with eye open was measured at 4 different luminous intensities. It was done to see that the EAR of a person remains approximately constant at all luminous intensity and also to show that the Raspberry Pi night vision camera works at different luminous intensity.

The following table shows the EAR of a person at the different luminous intensity:

Luminous intensity (lux)	Pictures showing the EAR
1.17	




13.26	
74.0	
209.2	

Table 06: EAR values at different luminous intensity

From the pictures above we can see that the EAR is 0.403, 0.395, 0.387, and 0.390. The average the EAR is 0.394. The difference between the EAR values at different light intensities is very less. Also the difference between the average EAR and the 4 different EAR is less.

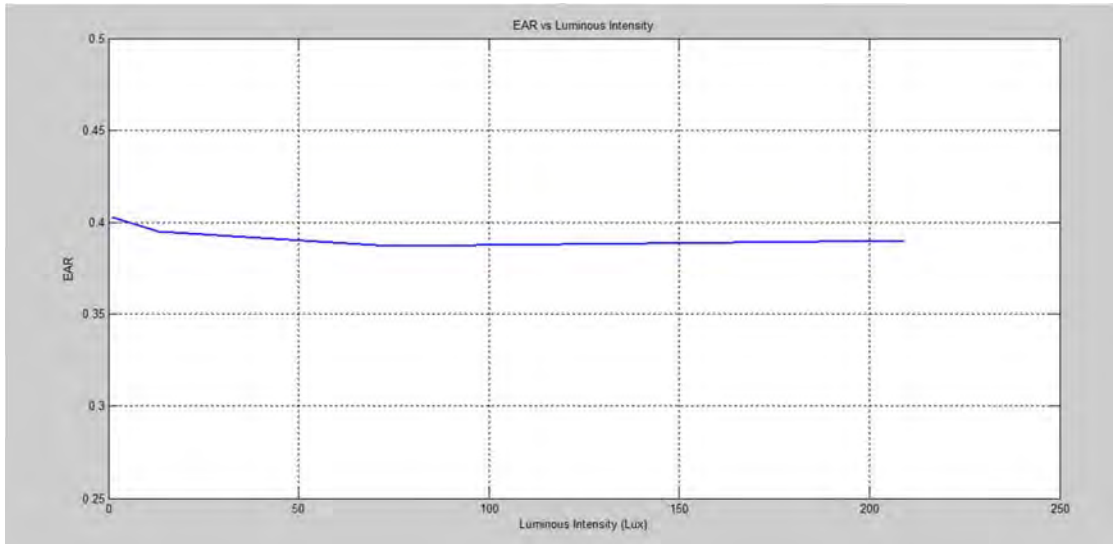


Figure 29: Graph for EAR vs Luminous intensity

The above graph also shows that EAR is almost constant for different luminous intensity. Thus, it can be concluded that the EAR is constant for a person in a particular situation (in this case eye open) and it does not depend on the light intensity. It can also be concluded that the night vision camera works at all light intensity.

5. Experiment to measure the heart rate from various people at different time of the day:

We carried out a test to measure the heart rate of various people at different time of the day. The table below shows the experimental values:

sample	Heart Rate (bpm)
1	79
2	93
3	82
4	67
5	90
6	96
7	81
8	77
9	112

10	92
11	108
12	74
13	93
14	90
15	88

Table 07: Heart rate of different people when they are awake at different time of the day

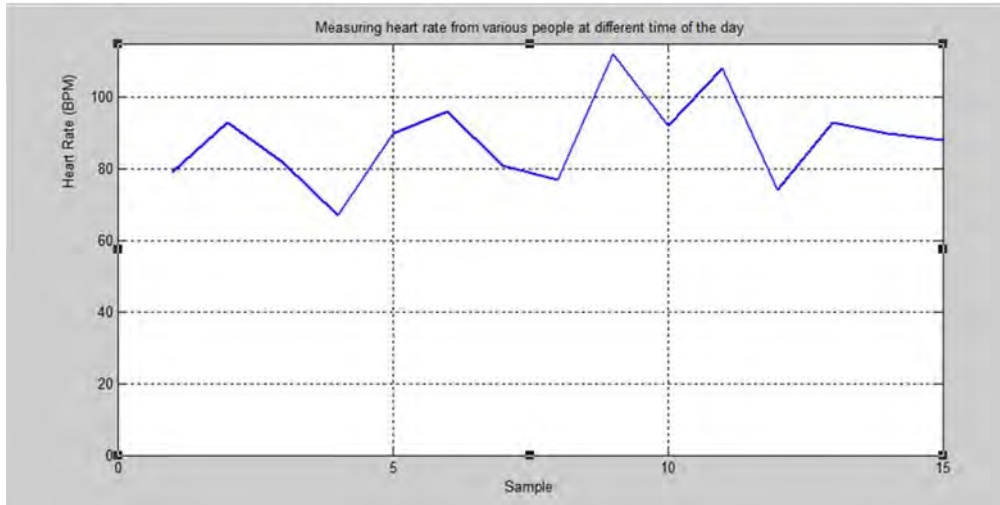


Figure 30: Graph of Heart rate vs Sample

The above graph shows that while awake the heart rate of an average healthy human does not drop below 60bpm. Hence, in our system 60bpm is set as the minimum threshold heart rate.

Conclusion

This chapter contains all the field tests we performed using our system. Using our experiments, we have found out the number of frames that should be set in the code for eye blink detection. We also found out the threshold EAR value. From our tests, we can conclude that the EAR of a person at any luminous intensity is constant in a particular situation(in this case eye open). We can also conclude that the night vision camera works at all luminous intensity.

CHAPTER 8

CONCLUSION

Introduction

This chapter contains brief summary of the entire system. It contains information about how the system was developed and how it works. This chapter also contains details of how we want to further develop our system to make it much more efficient.

8.1 Summary

The main objective of our work was to come up with a system that can detect driver's fatigue. After a lot of literature review, we found many methods to detect the fatigue level. But we decided to implement two methods that can be used in our system that we thought were the most accurate methods. Blink detection system and heart rate sensing are the two methods that we implemented in our project. Raspberry Pi was used as the processor. For eye blink detection the EAR was calculated and the groove ear clip was used to monitor the heart rate. These two methods together detected the drowsiness of drivers.

The developed system was tested under different conditions. The system can detect the EAR at different light intensity even in the dark as it used night vision camera. The system also accurately monitors the heartbeat. All the field test results prove that the system is quite efficient in monitoring the driver's fatigue.

The developed system is easy to develop, implement, use and gives accurate output. The efficiency of the system shows that it can be used for commercial purpose. But the system is just a prototype that needs certain modifications before it can be implemented in vehicles. The cost of the entire system is quite low which makes it affordable for use in commercial purpose.

8.2 Future Prospective

8.2.1 Steering wheel

Besides using facial detection method to detect the facial behavior of the driver such as eye movement, head tilting, blinking or even yawning heartbeat sensors could also be used to detect if the driver is falling asleep as we are using in our project. The limitation of using our system is that the sensor must be connected to the driver's earlobe or fingertip in order to

detect the driver’s heartbeat. And as our heartbeat sensor is not wireless a wire is always connected to the drivers body hence there would be some limitation of movement of the driver as well as it may cause the driver discomfort. In order to get rid of the problem a different type of heartbeat sensor could be built where no wire is required also the sensor need not to be attached to the driver’s body all the time. In the conference paper “*Wireless Sensor Embedded Steering Wheel For Real Time Monitoring Of Driver Fatigue Detection*” a method is introduced where the heartbeat sensor is embedded in the steering wheel of the car[24]. The Intelligent Steering Wheel Distributed Sensor Network (ISDSN) is the cluster of several sensors in the steering wheel [24]. The infrared emitter and receiver are placed in such a way so that very less space is consumed [24].

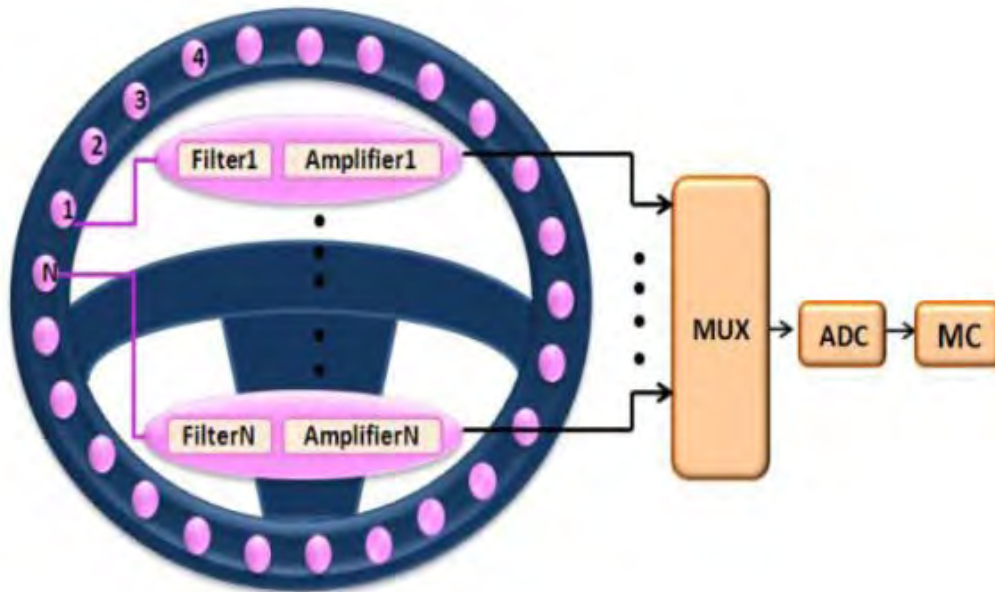


Figure 31: ISDSN Detection system (p. 224)

Source: Adapted from [24]

As the driver places his palm on the steering wheel the sensor could start working. The infrared ray is emitted from the emitter and it reflects back from the blood of the driver’s blood vessels. The volume of our blood flowing through the blood vessels changes according to our blood pressure[24]. Therefore the reflected ray received by the receiver gives the value of the driver’s heart rate. There are also filter to filter out the external noises from the original sensor’s value and amplifier to amplify the obtained value all incorporated within the steering

wheel[24].For further advancement a data storage device could also be added to keep the record of the data collected from the sensor [24].

In a journal “*Smartwatch-Based Driver Vigilance Indicator With Kernel-Fuzzy-C-Means-Wavelet Method*”a similar technique is used to determine the drivers fatigue stage. Here instead of using infrared ray ECG (electrocardiogram) sensors are used [25]. Moreover, the biomedical signals can be received by the smartwatch devices through BLE (Bluetooth low energy) [25]. Therefore this device could become wireless.

8.2.2 BSN, VANET AND WBAN

According to Association for Safe International Road Travel [26], each year about 1.3 million people die in road accidents, which means on an average 3287 people die every day on the roads. Needless to say, the importance of developing an integrated road safety system cannot be stressed enough. Along with the two sensors that we implemented in our system several other inputs from various sensors, such as Q sensor, alcohol sensor etc, can be taken to make the system more coherent and reliable in reducing such road fatalities and injuries.

Among several methods to prevent road accidents, Genaro Rebolledo-Mendez, Reyes, Paszkowicz, Domingo, and Skrypchuk (2014) [27] illustrated on developing a Body Sensor Network (BSN) by detecting the driver’s emotions. The two sensors used by Rebolledo-Mendez et al [27] are Q sensor for determining electrodermal activity (EDA) and NeuroSky’s Mindwave. Both these sensors are portable and readily available to buy. Q sensor measures skin conductance in microsiemens. The level of arousal of the driver can be predicted by this measurement. If the EDA value is high Rebolledo-Mendez et al [27] states that the person is more engaged, stressed or in a state of excitement, meaning higher level of arousal and similarly, disengagement, boredom or calmness can be concluded if the EDA value is low, indicating lower of arousal. On the other hand, NeuroSky’s Mindwave determines whether the driver is in the state of attention or meditation. These two types of neural activities can be determined by Beta waves and Alpha waves. Rebolledo-Mendez et al [27] explains an increase in Beta waves indicates a state of attention meaning the driver is aware and similarly, increase in Alpha waves indicates a state meditation. According to Rebolledo-Mendez et al [27], data is not stored on the NeuroSky but instead it has external storage mechanism with Bluetooth that communicates with the On Board Unit (OBU) of the vehicle. From their test results, Rebolledo-Mendez et al [27] derives the correlation between physiological responses and emotional information given by 13 drivers themselves using

logical regression. Their results show high coefficients of variance between the neural activity predicted by NeuroSky's Mindwave and self reported data given by the drivers. Whereas, the coefficient of variance is very low between EDA results and data reported about their emotions while driving by the drivers themselves.

From our literature review, we found another reason for road accidents is drunk driving. According to National Highway Traffic Safety Administration, in 2016, 10497 died due to accidents caused by drunken driving [28]. According to WHO report, small amount of alcohol concentration in blood can lead to accidents but the risk of accidents increase when the blood alcohol concentration, $BAC \geq 0.04g/dl$ [29].

In our system we can incorporate an alcohol sensor that would continuously monitor the blood alcohol concentration. An infrared breath analyzer can be placed on the steering wheel. The breath analyzer will detect the driver's exhaled breath and will monitor the BAC. We can set the threshold BAC to $0.04g/dl$ and if the detected BAC is greater than $0.04g/dl$ then it will give a signal. We can connect a buzzer at the output which will buzz if $BAC > 0.04g/dl$.



Figure 32: Placement of alcohol sensor on steering wheel

The integration of Body Sensor Network and Vehicular Ad-hoc Networks can be used to reduce accidents [30]. The BSN detects four psychological states. Firstly, it detects the drowsiness of drivers that we have already implemented in our project. But [30] talks about measuring the heart rate using ECG sensor attached to the steering wheel. In future, we can also replace the groove ear clip with the ECG sensor and drivers would not have to wear the groove ear clip in their finger or ear lobe which would make the system more convenient to

use. Secondly, BSN detects the drunken state of drivers using alcohol sensors. Thirdly, BSN uses ECG, EDA sensors to detect the emotional state of the drivers. Fourthly, BSN detects distracted driving motion sensors, accelerometers, gyroscopes.



Figure 33: Proposed Scenario of BSN

Source: Adapted from [30]

All the data obtained from the BSN goes to the monitoring station for processing. A processing node which can be a Personal Digital Assistant (PDA), smartphone, laptop, microcontroller, Field Programmable Gate Array (FPGA) The monitoring station has two modules: one for extraction of the features from the sensors and the other is the intelligent driver's state recognition module to determine if the driver has one of the four psychological states [30]. From the monitoring station the data goes the car's OBU and the OBU has an alarm notification module. If it detects one of the four psychological features it will forward message to the VANET or the RSU. For this wireless communication means like Wi-Fi, GPS is required. Vehicular Ad-hoc Network as described in [31] is a network of nodes on the road where each vehicle is a node. All vehicles within a certain range, for example 100 to 300 meters, are connected through devices such as personal digital assistants (PDA), smart

phones, or laptops etc. Thus a network is created with numerous vehicles continuously adding and falling out from it. The network also contains Road Side Units (RSU) which is placed in fixed intervals in areas where the traffic is low. Therefore, the three kinds of connections with a VANET system described in [31] are Vehicle to Vehicle (V2V), Vehicle to Roadside (V2R) and Vehicle to infrastructure (V2I). The possible future position of each vehicle can hence be predicted if the road map, road structure and the direction and speed of the vehicle is known.

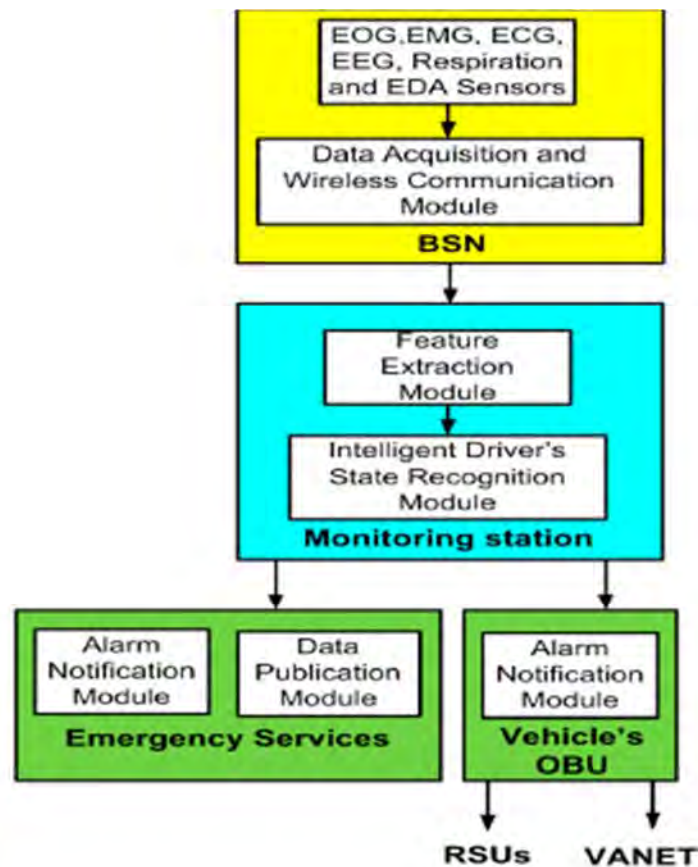


Figure 34: The model for future work

Source: Adapted from [30]

A hybrid VANET model is illustrated in [31] using Wireless Body Area Network (WBAN). Wireless sensors are connected in the body to form WBAN network to increase the safety of

careless pedestrians. Flexible, less power consuming, compact devices such as, Google glasses, smart oars or smart watches etc can worn by the pedestrians to form this network. In [32] this hybrid model is introduced keeping in mind pedestrians in black spots which the driver cannot see. Furthermore, the safety of disabled, handicapped, people with poor vision, drunken pedestrians and children on the streets can be increased drastically if they wear a wearing a device which can communicate with the RSU through Wi-Fi communication interface and alert drivers of their approach and take necessary action such as, slowing down, applying brakes or changing lanes, in order to avoid any imminent accidents.



Figure 35: System Model of WBAN

Source: Adapted from [31]

Experiments using GrooveNet simulator was done assuming vehicles going on a model of 3-lane highway and results in [31] show drastic improvement in hybrid VANET model. When the speed of the vehicle is more than 36km/hr, then only it was a danger for pedestrians in the hybrid VANET model, where even vehicles driven at a speed of 11km/hr were risky to pedestrians when WBAN was not used.



Figure 36: WBAN communication alerting the vehicle

Source: Adapted from [31]

8.2.3 Limitation

Raspberry Pi 3 Model B has a maximum CPU speed of 1.2 GHz and that limits the number of tasks we can perform on it. When the program is run to detect the eyes and measure heart rate, the CPU of the Raspberry Pi runs upto 80% of its maximum processing power. For future prospects, we discussed that an alcohol sensor can be introduced in the system to reduce drunk driving and heartbeat can be measured through the use of a band that send data wirelessly to the micro controller. As more sensors are introduced, the system becomes more complex and more powerful CPU will be required to do the tasks. Thus we will need to change the microcontroller when making the system more advanced.

8.2.4 Costing

The table below shows all the components we needed to build up the system along with their costing.

Components	PRICE(Tk.)	PRICE(USD)
Raspberry Pi 3 model B	3950	49.375
5 Megapixel night vision for Raspberry Pi	4351	54.388
Raspberry Pi case	556	6.950
Grove ear clip heart rate sensor	2351	29.388
Bread board	105	1.313
MCP3008	361	4.513
Buzzer	15	0.188
LED	2	0.025
Remax Car charger	700	8.750
Jumper wires	33	0.413
USB Cable	90	1.125
16 Memory Card	900	11.25
Total	13414	159

Table 08: Costing of the system

Conclusion

In this chapter, we summarized all the things we did to develop our system, how the system works and its limitations. In this system we used only two sensors to detect the driver's drowsiness. But in future we want to incorporate more sensors to form a BSN that will be more efficient in reducing the road accidents.

REFERENCE

1. The Daily Star. 'Road Accidents: Sharp rise in fatalities'. (2017). Available: <https://www.thedailystar.net/backpage/road-accidents-sharp-rise-fatalities-1426999>[Accessed 16 Apr. 2018].
2. UCLA Health. 'Drowsy Driving'. [Online]. Available: <http://sleepcenter.ucla.edu/drowsy-driving> [Accessed: 10-April-2018]
3. Real Time Eye Detection and Tracking Method for Driver Assistance System
By Sayani Ghosh, Tanaya Nandy and Nilotpal Manna. Retrieved from http://www.springer.com/cda/content/document/cda_downloaddocument/9788132222552-c2.pdf?SGWID=0-0-45-1492990-p177196737
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, Jan 2016 Retrieved from https://www.ijirset.com/upload/2016/january/70_15_SMART.pdf
5. TECHSHOP bd.com. [Online]. Available: <https://www.techshopbd.com/product-categories/boards/1253/arduino-uno-r3-china-techshop-bangladesh> [Accessed: 10-April-2018]
6. Adafruit.com, 'Raspberry Pi Analog to Digital Converter', 2016. [Online]. Available: <https://learn.adafruit.com/raspberry-pi-analog-to-digital-converters/mcp3008>. [Accessed: 07-April-2018].
7. Sparkfun.com, 'Serial Peripheral Interface (SPI)', [Online]. Available: <https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi>. [Accessed: 07-April-2018].
8. Seeedstudio.com, 'Grove-Ear-clip Heart Rate Sensor', 2013. [Online]. Available: http://wiki.seeed.cc/Grove-Ear-clip_Heart_Rate_Sensor/. [Accessed: 07-April-2018].
9. Livescience.com, 'What is a normal heart rate?' 2017. [Online]. Available: <https://www.livescience.com/42081-normal-heart-rate.html>. [Accessed: 07-April-2018].
10. Medicalnewstoday.com, 'What should my heart rate be?', 2017. [Online]. Available: <https://www.medicalnewstoday.com/articles/235710.php>. [Accessed: 07-April-2018].
11. Livestrong.com, 'Causes of Fatigue and a racing heart', 2017. [Online]. Available: <https://www.livestrong.com/article/258899-causes-of-fatigue-and-a-racing-heart/>. [Accessed: 07-April-2018].

12. livestrong.com, 'What Is a Normal Heart Rate While Sleeping?', 2017, [Online]. Available:<https://www.livestrong.com/article/105256-normal-heart-rate-sleeping/>. [Accessed: 07-April-2018].
13. Chonowski, K. (2017, December 26). Top Ten Things to Know About the Raspberry Pi 3. Retrieved from <https://www.arrow.com/en/research-and-events/articles/top-ten-things-to-know-about-the-raspberry-pi-3>[Accessed: 26-February-2018]
14. Raspberry Pi 3 is out now! Specs, benchmarks & more. (2017, June 02). Retrieved from <https://www.raspberrypi.org/magpi/raspberry-pi-3-specs-benchmarks/>[Accessed: 26-February-2018]
15. Raspberry Pi GPIO Pinout. (n.d.). Retrieved from <https://pinout.xyz/>[Accessed: 26-February-2018]
16. Raspberry Pi FAQs - Frequently Asked Questions. (n.d.). Retrieved from <https://www.raspberrypi.org/help/faqs/#powerReqs>[Accessed: 26-February-2018]
17. Best Operating Systems for Raspberry Pi 2 & 3 | Top 10 OS List. (2017, September 20). Retrieved from <https://www.raspberrypistarterkits.com/products/operating-systems-raspberry-pi/>[Accessed: 26-February-2018]
18. ModMyPi | Raspberry Pi Camera Board - Night Vision & Adjustable-Focus Lens (5MP). (n.d.). Retrieved from <https://www.modmypi.com/raspberry-pi/camera/camera-boards/raspberry-pi-night-vision-camera>[Accessed: 26-February-2018]
19. Soukupova, T., & Cech, J. (2016). Real-Time Eye Blink Detection using Facial Landmarks. Retrieved from <https://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pdf>.
20. <https://www.pyimagesearch.com/2017/10/23/raspberry-pi-facial-landmarks-drowsiness-detection-with-opencv-and-dlib/>
21. tutorials-raspberrypi.com, 'Raspberry Pi heartbeat/pulse measuring', [Online]. Available:<https://tutorials-raspberrypi.com/raspberry-pi-heartbeat-pulse-measuring/>. [Accessed: 07-April-2018].
22. github.com, 'Pulse sensor showing BPM 150+', 2017, [Online]. Available: https://github.com/WorldFamousElectronics/PulseSensor_Amped_Arduino/issues/60. [Accessed: 07-April-2018].

23. github.com, 'Detecting BPM without touching the sensor', 2015, [Online]. Available:https://github.com/WorldFamousElectronics/PulseSensor_Amped_Arduino/issues/13 . [Accessed: 07-April-2018].
24. B. Thomas, and G. Ashutosh, "Wireless Sensor Embedded Steering Wheel For Real Time Monitoring Of Driver Fatigue Detection" in Proc. of the Intl. Conf. on Advances in Computer Science and Electronics Engineering, 2014
25. B.G.Lee, Jae-H. Park, C.C.Pu, and Wan-Y.Chung, "Smartwatch-Based Driver Vigilance Indicator WithKernel-Fuzzy-C-Means-Wavelet Method" in IEEE sensors journal, vol.16 ,no.1 ,Jan.2016
26. <http://asirt.org/initiatives/informing-road-users/road-safety-facts/road-crash-statistics>[Accessed: 11-April-2018].
27. G. Rebolledo-Mendez, A. Reyes, S. Paszkowicz, M. C. Domingo, and L. Skrypchuk, "Developing a Body Sensor Network to Detect Emotions During Driving," in IEEE transaction on intelligent transport systems, vol. 15, no. 4, Aug. 2014
28. National Highway Traffic Safety Administration. Drunk Driving. [online]. Available: <https://www.nhtsa.gov/risky-driving/drunk-driving> [Accessed 11-April-2018]
29. World Health Organization. Road Traffic Injuries. (2018). Available [Online]: <http://www.who.int/mediacentre/factsheets/fs358/en/> [Accessed: 11-April-2018]
30. Angelica Reyes-Muñoz 1*, Mari Carmen Domingo 2, Marco Antonio López-Trinidad 3 and José Luis Delgado 3 "Integration of Body Sensor Networks and Vehicular Ad-hoc Networks for Traffic Safety". Retrieved from <http://www.mdpi.com/1424-8220/16/1/107>
31. D. Sam, E. Evangelin, and Dr. V. Cyril Raj, "A novel idea to improve pedestrian safety in Black Spots using a Hybrid VANETof vehicular and body sensors" in International Conference on Innovation Information in Computing Technologies, 2015

APPENDIX

a. Source code for eye blink detection and heart rate detection using Raspberry Pi, see [20]

```
# USAGE
# python pi_detect_drowsiness.py --cascade
haarcascade_frontalface_default.xml --shape-predictor
shape_predictor_68_face_landmarks.dat
# python pi_detect_drowsiness.py --cascade
haarcascade_frontalface_default.xml --shape-predictor
shape_predictor_68_face_landmarks.dat --alarm 1

# import the necessary packages
import sys
import os
from math import fabs
#sys.path.append('/home/pi/Adafruit_Python_MCP3008/examples/')
from pulsesensor import Pulsesensor

from imutils.video import VideoStream
from imutils import face_utils
import numpy as np
import argparse
import imutils
import time
import dlib
import cv2
import RPi.GPIO as GPIO
from time import sleep

#-----CONFIGURATION FOR HEARTBEAT SENSOR-----
-----

p = Pulsesensor() #The class that reads the value from ADC
p.startAsyncBPM() #Start reading the value periodically in a different
thread

buzzer_pin=18 #Pin number to which the buzzer is connected [BCM,
Raspberry Pi Model B]
led_pin=14 #Pin number to which the LED light is connected. It remains
on when the eyes are closed

min_bpm = 50 #Minimum BPM threshold below which the buzzer will sound
max_bpm = 200 #mum BPM threshold above which the buzzer will sound

#This parameters filter noise value by accepting fixed number of BPM
values
#which exists within a valid range

last_bpm = 0 #The Last BPM read from the Sensor class
bpm_in_range = 0 #Number of BPM value detected which is within the
range
bpm_max_val = 3 #Number of consecutive BPM values that must exist
within a valid range
```

```

#Maximum difference between last value read and current value read
range_val = 20 #Maximum difference between last and current BPM value
that validates range
processing = False #The detected BPM value was obtained while
calculating BPM or from the sensor

#-----CONFIGURATION FOR HEARTBEAT SENSOR-----
-----

GPIO.setmode(GPIO.BCM)
GPIO.setup(24, GPIO.OUT)
GPIO.setup(buzzer_pin,GPIO.OUT)
GPIO.setup(led_pin, GPIO.OUT)

#-----FUNCTIONS OF HEARTBEAT SENSOR-----
-----
#-----Functions for controlling the buzzer-----
def cleanup():
GPIO.cleanup()
def buzzer_on():
if GPIO.input(buzzer_pin) == 0:
GPIO.output(buzzer_pin, GPIO.HIGH)
time.sleep(1)
def buzzer_off():
if GPIO.input(buzzer_pin) == 1:
GPIO.output(buzzer_pin, GPIO.LOW)

def light_on():
if GPIO.input(led_pin) == 0:
GPIO.output(led_pin, GPIO.HIGH)
time.sleep(1)
def light_off():
if GPIO.input(led_pin) == 1:
GPIO.output(led_pin, GPIO.LOW)
#-----FUNCTIONS OF HEARTBEAT SENSOR-----
-----

#-----Test Buzzer-----
#print "TESTING BUZZER. TURNING ON THE BUZZER"
#GPIO.output(buzzer_pin, GPIO.HIGH)
#time.sleep(2)
#print "TESTING BUZZER. TURNING OFF THE BUZZER"
#GPIO.output(buzzer_pin, GPIO.LOW)

def euclidean_dist(ptA, ptB):
    # compute and return the euclidean distance between the two
    # points
    return np.linalg.norm(ptA - ptB)

def eye_aspect_ratio(eye):
    # compute the euclidean distances between the two sets of
    # vertical eye landmarks (x, y)-coordinates
    A = euclidean_dist(eye[1], eye[5])
    B = euclidean_dist(eye[2], eye[4])

    # compute the euclidean distance between the horizontal

```



```

    # eye landmark (x, y)-coordinates
    C = euclidean_dist(eye[0], eye[3])

    # compute the eye aspect ratio
    ear = (A + B) / (2.0 * C)

    # return the eye aspect ratio
    return ear

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-c", "--cascade", required=True,
                help = "path to where the face cascade resides")
ap.add_argument("-p", "--shape-predictor", required=True,
                help="path to facial landmark predictor")
ap.add_argument("-a", "--alarm", type=int, default=0,
                help="boolean used to indicate if TraffHat should be used")
args = vars(ap.parse_args())

# check to see if we are using GPIO/TrafficHat as an alarm
if args["alarm"] > 0:
    from gpiozero import TrafficHat
    th = TrafficHat()
    print("[INFO] using TrafficHat alarm...")

# define two constants, one for the eye aspect ratio to indicate
# blink and then a second constant for the number of consecutive
# frames the eye must be below the threshold for to set off the
# alarm
EYE_AR_THRESH = 0.2
EYE_AR_CONSEC_FRAMES = 6

# initialize the frame counter as well as a boolean used to
# indicate if the alarm is going off
COUNTER = 0
ALARM_ON = False

# load OpenCV's Haar cascade for face detection (which is faster than
# dlib's built-in HOG detector, but less accurate), then create the
# facial landmark predictor
print("[INFO] loading facial landmark predictor from " +
      str(args["cascade"]))
detector = cv2.CascadeClassifier(args["cascade"])
predictor = dlib.shape_predictor(args["shape_predictor"])

# grab the indexes of the facial landmarks for the left and
# right eye, respectively
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]

# start the video stream thread
print("[INFO] starting video stream thread...")
#vs = VideoStream(src=0).start()
vs = VideoStream(usePiCamera=True).start()
time.sleep(1.0)

# loop over frames from the video stream
while True:

```

```

print "Running"
light_off()
buzzer_off()
    # grab the frame from the threaded video file stream, resize
    # it, and convert it to grayscale
    # channels)
    #time.sleep(0.1)
frame = vs.read()
frame = imutils.rotate(frame, 180)
frame = imutils.resize(frame, width=450)
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
try:
    # detect faces in the grayscale frame
    rects = detector.detectMultiScale(gray, scaleFactor=1.1,
        minNeighbors=5, minSize=(30, 30),
        flags=cv2.CASCADE_SCALE_IMAGE)
    # loop over the face detections
    for (x, y, w, h) in rects:
        # construct a dlib rectangle object from the Haar cascade
        # bounding box
        rect = dlib.rectangle(int(x), int(y), int(x + w),
            int(y + h))

        # determine the facial landmarks for the face region, then
        # convert the facial landmark (x, y)-coordinates to a NumPy
        # array
        shape = predictor(gray, rect)
        shape = face_utils.shape_to_np(shape)

        # extract the left and right eye coordinates, then use the
        # coordinates to compute the eye aspect ratio for both eyes
        leftEye = shape[lStart:lEnd]
        rightEye = shape[rStart:rEnd]
        leftEAR = eye_aspect_ratio(leftEye)
        rightEAR = eye_aspect_ratio(rightEye)

        # average the eye aspect ratio together for both eyes
        ear = (leftEAR + rightEAR) / 2.0

        # compute the convex hull for the left and right eye, then
        # visualize each of the eyes
        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)

        # check to see if the eye aspect ratio is below the blink
        # threshold, and if so, increment the blink frame counter
        if ear < EYE_AR_THRESH:
            COUNTER += 1

            # if the eyes were closed for a sufficient number of
            # frames, then sound the alarm
            if COUNTER >= EYE_AR_CONSEC_FRAMES:
                # if the alarm is not on, turn it on
                if not ALARM_ON:
                    ALARM_ON = True

```

```

should
    # check to see if the TrafficHat buzzer
    # be sounded
    ifargs["alarm"] > 0:
        th.buzzer.blink(0.1, 0.1, 10,
            background=True)

        # draw an alarm on the frame
        cv2.putText(frame, "DROWSINESS ALERT!", (10,
30), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
        #GPIO.output(24, 1)
print "-----EYES CLOSED!!!!!!!!!!!!!!!!!!!!-----"
        #-----DETECTION OF
HEARTBEAT PULS-----
        #buzzer_off()

light_on()
bpm = p.BPM
if bpm == 0:
    processing = False
else:
    processing = True
print "Difference: " + str(fabs(bpm - last_bpm))
iffabs(bpm-last_bpm) <= range_val:
    ifbpm_in_range>= bpm_max_val:
        bpm_in_range = bpm_max_val - 1
    else:
        bpm_in_range = bpm_in_range + 1
    bpm = 0
elif bpm != 0:
    last_bpm = bpm
    bpm_in_range = 0
    bpm = 0

        p.BPM = 0
print "Last BPM: " + str(last_bpm)
print "No. of BPM in range: " + str(bpm_in_range)
if bpm > 0:
    print("-----BPM: %d-----" % bpm)
    if bpm <min_bpm or bpm >max_bpm:
        buzzer_on()
    else:
        if processing:
            print("-----Calculating BPM-----")
        else:
            print("-----No Heartbeat found-----")
        processing = False

        #-----DETECTION OF HEARTBEAT PULSE-
-----

        # otherwise, the eye aspect ratio is not below the blink
        # threshold, so reset the counter and alarm
        else:
            COUNTER = 0
            ALARM_ON = False
            #GPIO.output(24, 0)
            #light_off()

```

```

        # draw the computed eye aspect ratio on the frame to help
        # with debugging and setting the correct eye aspect ratio
        # thresholds and frame counters
        cv2.putText(frame, "EAR: {:.3f}".format(ear), (300, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

    # show the frame
    cv2.imshow("Frame", frame)
    # if the `q` key was pressed, break from the loop
exceptKeyboardInterrupt:
    print "Exiting"
    break
        #vs.stop()
        #os._exit(1)
except:
    print "Error"
        #vs.stop()
        #os._exit(1)
break
    key = cv2.waitKey(1) & 0xFF

    # if the `q` key was pressed, break from the loop
    if key == ord("q"):
        break

# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
os._exit(1)

```

b. Source code for heart beat detection using Arduino, see [8]

```
// Function: This program can be used to measure heart rate, the lowest
pulse in the program be set to 30.

// Use an external interrupt to measure it.

// Hardware: Grove - Ear-clip Heart Rate Sensor, Grove - Base Shield,
Grove - LED

// Arduino IDE: Arduino-1.0

// Author: FrankieChu

// Date: Jan 22, 2013

// Version: v1.0

// by www.seeedstudio.com

#define LED 11//indicator, Grove - LED is connected with D4 of Arduino
boolean led_state = LOW;//state of LED, each time an external interrupt
//will change the state of LED

unsigned char counter;
unsigned long temp[21];
unsigned long sub;
booldata_effect=true;
unsigned intheart_rate;//the measurement result of heart rate

const int max_heartpluse_duty = 2000;//you can change it follow your
system's request.

//2000 means 2 seconds. System return error
//if the duty overtrip 2 second.

void setup()
{
pinMode(LED, OUTPUT);
Serial.begin(9600);
Serial.println("Device Starting...");
delay(5000);
arrayInit();
Serial.println("Heart rate test begin.");
attachInterrupt(0, interrupt, RISING);//set interrupt 0,digital port 2
```

```

}
void loop()
{
    //digitalWrite(LED, led_state);//Update the state of the indicator
}
/*Function: calculate the heart rate*/
void sum()
{
    if(data_effect)
    {
        heart_rate=1200000/(temp[20]-temp[0]);//60*20*1000/20_total_time
        Serial.print("Heart_rate_is:\t");
        Serial.println(heart_rate);
        if(heart_rate< 85)
        {
            Serial.println("Buzzer ON");
            digitalWrite(LED, HIGH);
            delay(1000);
        }
        else
        {
            Serial.println("Buzzer OFF");
            digitalWrite(LED, LOW);
        }
    }
    data_effect=1;//sign bit
}
/*Function: Interrupt service routine.Get the sigal from the external
interrupt*/
void interrupt()
{
    temp[counter]=millis();
}

```

```

Serial.println(counter,DEC);
    //Serial.println(temp[counter]);
switch(counter)
{
case 0:
sub=temp[counter]-temp[20];
    //Serial.println(sub);
break;
default:
sub=temp[counter]-temp[counter-1];
    //Serial.println(sub);
break;
}
if(sub>max_heartpluse_duty)//set 2 seconds as max heart pluse duty
{
data_effect=0;//sign bit
counter=0;
Serial.println("Heart rate measure error,test will restart!" );
arrayInit();
}
if (counter==20&&data_effect)
{
counter=0;
sum();
}
else if(counter!=20&&data_effect)
counter++;
else
{
counter=0;
data_effect=1;
}

```

```
}  
/*Function: Initialization for the array(temp)*/  
voidarrayInit()  
{  
for(unsigned char i=0;i < 20;i ++)  
    {  
temp[i]=0;  
    }  
temp[20]=millis();  
}
```