P.E.S COLLEGE OF ENGINEERING,

MANDYA-571401

(An Autonomous Institute under Visvesvaraya Technological University, Belgaum)

A Project Report

On

"Driver Drowsiness Detection Using Raspberry Pi"

Submitted in partial fulfilment of the requirement
for the award of the
BACHELOR OF ENGINEERING DEGREE

Submitted by

MD EHSAN AHSAN	[USN:4PS15CS046]
SRIMAN MISHRA	[USN:4PS15CS112]
NITISH RANJAN	[USN:4PS15CS068]
ANUJ SINHA	[USN:4PS15CS012]

Under the guidance of

T.M RAGHAVENDRA BABU
Asst. Professor

Department of Computer Science and Engineering
P.E.S. College of Engineering, Mandya.

2019-2020

P.E.S COLLEGE OF ENGINEERING,

MANDYA-571401

(An Autonomous Institute under Visvesvaraya Technological University, Belgaum)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Certificate

This is to certify that, **MD EHSAN AHSAN [USN:4PS15CS046], SRIMAN MISHRA [USN:4PS15CS112], NITISH RANJAN [USN:4PS15CS068], ANUJ SINHA [USN:4PS15CS012]** student of eight semester in computer science and Engineering, P.E.S College of Engineering, Mandya, has satisfactorily completed the report on **"Driver Drowsiness Detection Using Raspberry Pi"** during the year 2019-2020. This project report satisfies the academic requirements in respect to the prescribed eight semester of computer Science & Engineering discipline.

Signature of the Guide	**Signature of the HOD**
T.M RAGHAVENDRA BABU	**Dr. MC.PADMA**
Asst. Professor	**Professor and Head,**
Department of CS&E,	Department of CS&E,
PESCE, MANDYA	PESCE, MANDYA

Signature of Examiner:

1. _____

2. _____

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible because ***"Success is the abstract of hard work and perseverance, but steadfast of all is encouraging guidance"***. So, we acknowledge all those whose guidance and encouragement served as a beacon light and crowned our effort with success.

We whole-heartedly express our sincere thanks to Dr. H.V.RAVINDRA, Principal PESCE, Mandya, for having supported us in our academic endeavors.

We would like to thank our college administration for providing a conductive environment and also suitable facilities for this project. We would like to thank **Dr. M.C.PADMA**, Head of the Department of CS&E, PESCE Mandya for providing the inspiration requires for taking the project to its completion. It is a great pleasure to thank our guide teacher in-charge **T.M RAGHAVENDRA BABU,** Asst. Professor, Department of CS&E, PESCE Mandya for his constant encouragement, guidance and support throughout this project.

We thank all the staff members of the department of CS&E for providing resources for the completion of the project.

Finally, we thank all of those who have contributed directly or indirectly in making this project a Grand Success

ABSTRACT

Driver drowsiness is a major cause of several highway calamities leads to severe physical injuries, loss of human life. The implementation of driver drowsiness detection in real-time would aid in avoiding major accidents. The system is designed for vehicle where the driver's fatigue or drowsiness is detected and alerts the driver and passengers. Based on computer vision techniques, the driver's face is located from a video captured in the vehicle. Then, face detection is employed to locate the regions of the driver's eyes, which are used as the templates for eye tracking in subsequent frames. On the detection of drowsiness, the programmed system cautions the driver through an alarm to ensure vigilance. The proposed method constitutes of various stages to determine wakefulness of the driver. A warning alarm will be generated if the driver is sleepy. HaarCascade Classifiers is used to detect the blink duration of the driver and Eye Aspect Ratio (EAR) is calculated. The proposed system will be tested on a Raspberry pi 4 Model with 2GB RAM with use of HD Webcam. Thus, it can be concluded that the proposed approach will be low cost and effective solution method for a real-time of driver drowsiness detection.

Keywords: Eye Aspect Ratio, Raspberry Pi, Webcam.

CONTENTS

CHAPTER **Page no.**

CHAPTER 1

INTRODUCTION

Drowsy driving is one of the major causes behind fatal road accidents. One of the recent studies shows that one out of five road accidents are caused by drowsy driving which is roughly around 21% of road accidents, and this percentage is increasing every year as per global status report on road safety 2018, based on the data from 180 different countries. This certainly highlights the fact that across the world the total numbers of road traffic deaths are very high due to driver's drowsiness. Driver fatigue, drink-and-drive and carelessness are coming forward as major reasons behind such road accidents. Many lives and families are getting affected due to this across various countries. Real time drowsy driving detection is one of the best possible majors that can be implemented to assist drivers to make them aware of drowsy driving conditions. Such driver behavioral state detection system can help in catching the driver drowsy conditions early and can possibly avoid mishaps.

Among these the major cause is due to driver errors and recklessness. Driver fatigue is cause behind such mishaps. Heavy traffic, increasing automotive population, adverse driving conditions, tight commute time requirements and the work loads are few major reasons behind such fatigue. With this, we are presenting technique to detect driver drowsiness using of Open CV, raspberry pi and image processing. Several studies have shown various possible techniques that can detect the driver drowsiness. Such driver drowsiness detection can be measured using physiological measures, ocular measure and performance measure. Among these physiological measure and ocular measure can give more accurate results. Physiological measure includes brain waves, heart rate, pulse rate measurements and these requires some sort of physical connection with the driver such as connecting electrode to the driver body. But this leads to discomfort in driving conditions. But ocular measure can be done without physical connection. Ocular measure to detect driver eye condition and possible vision based on eye closure is well suited for real world driving conditions, since it can detect the eyes open/ closed state non-intrusively using a camera. In real time Driver Drowsiness System using

Image Processing, capturing drivers eye state using computer vision-based drowsiness detection systems have been done by analyzing the interval of eye closure and developing an algorithm to detect the driver's drowsiness in advance and to warn the driver by in vehicles alarm.

CHAPTER 2

LITERATURE SURVEY

[1.] While significant research has been done into the physiological mechanisms that underlie sleep and the sleep/wake cycle, the available data regarding the nature of drowsiness is far more limited. An objective measurement of drowsiness would have clinical utility, and a precise definition of the drowsy state could offer insights into the nature and purpose of sleep. Studies utilizing fMRI have demonstrated increased area of central nervous system involvement with tasks of increasing complexity. Preliminary data from studies of magnetoencephalography (MEG) with a receptive language task have demonstrated a progressive increase in global coherence of activity between MEG sensors with increasing drowsiness. The relationship between global coherence and the level of drowsiness suggests that the former may serve as an objective measurement of the latter. If true, the relationship suggests the hypothesis that drowsiness may be defined as a progressive loss of cortical network processing efficiency, requiring the recruitment of greater amounts of cortical tissue to perform the same task.

[2.]Drowsy driving is a serious problem that leads to thousands of automobile crashes each year. This report, sponsored by the National Center on Sleep Disorders Research (NCSDR) of the National Heart, Lung, and Blood Institute of the National Institutes of Health, and the National Highway Traffic Safety Administration (NHTSA), is designed to provide direction to an NCSDR/NHTSA educational campaign to combat drowsy driving. The report presents the results of a literature review and opinions of the Expert Panel on Driver Fatigue and Sleepiness regarding key issues involved in the problem.

[3.]Drivers in the United States report having fallen asleep or nodded off while driving at alarming rates: more than two in five (43.2%) have fallen asleep or nodded off while driving at some point in their lives, including 2.5% who did so within the past month, 6.6% within the past six months, and 10.0% within the past year. These results are highly consistent with past surveys conducted by the AAA Foundation, NHTSA, and the CDC:

estimates of lifetime prevalence of falling asleep at the wheel ranged from 37% to 41%, while estimates of past month drowsy driving converged at 4% (CDC, 2013; Royal, 2003; Tefft, 2010).

[4.] Over 1.2 million people die each year on the world's roads, and between 20 and 50 million suffer non-fatal injuries. In most regions of the world this epidemic of road traffic injuries is still increasing. In the past five years most countries have endorsed the recommendations of the World report on road traffic injury prevention which give guidance on how countries can implement a comprehensive approach to improving road safety and reducing the death toll on their roads. To date, however, there has been no global assessment of road safety that indicates the extent to which this approach is being implemented. This Global status report on road safety is the first broad assessment of the status of road safety in 178 countries, using data drawn from a standardized survey conducted in 2008. The results provide a benchmark that countries can use to assess their road safety position relative to other countries, while internationally the data presented can collectively be considered as a global "baseline" against which progress over time can be measured.

[5.] The Department for Transport has identified the need for a critical evaluation of up to date evidence to accurately determine the current scale of the driver fatigue problem in the UK, and to highlight evidence gaps that may require further research.

A consortium consisting of Clockwork Research, the University of Leeds and the Transport Research Laboratory (TRL) have worked together to produce a report that meets the objectives of the Department for Transport; namely, to provide a comprehensive and critical review of the literature that synthesizes the evidence relating to fatigue and road safety. The report is intended to provide guidance for the Department for Transport Research, Policy and THINK! communication teams.

[6.] The purpose of this study was to quantify the relationship between recent sleep and relative risk of involvement in a motor vehicle crash. The measure of recent sleep used for this study was the total amount that a driver had slept in the previous 24 hours. The risk of crash involvement associated with a given amount of sleep relative to the risk associated with the reference amount of sleep (7+ hours) was estimated by comparing the amount of sleep reported by drivers who on-scene crash investigators found to have committed an unsafe or illegal action, inaction, or error that was found to be the critical reason for the crash, to the amount of sleep reported by drivers who investigators found not to have contributed to the crash in such a manner. Assuming that drivers involved in crashes to which they did not contribute comprise a representative sample of all drivers on the road, the extent to which a given amount of sleep is more (or less) prevalent among drivers who contributed to crashes than among drivers involved in crashes to which they did not contribute reflects the extent to which drivers who had slept for that amount are over-involved (or under-involved) in crashes compared with the reference amount. The data analyzed for this study was from the National Highway Traffic Safety Administration's (NHTSA's) National Motor Vehicle Crash Causation Survey (NMVCCS), which comprised a representative sample of police-reported crashes that occurred between July 2005 and December 2007.

[7.] We develop a new approach for detecting drowsiness-related lane departures using steering wheel angle data that employ an ensemble definition of drowsiness and a random forest algorithm. Data collected from 72 participants driving the National Advanced Driving Simulator are used to train and evaluate the model. The model's performance was assessed relative to a commonly used algorithm, percentage eye closure (PERCLOS).

[8.] The purpose of this research is to summarize the results of a three-year study in which the main objective was to develop reliable algorithms for the detection of driver impairment due to drowsiness. More specifically, the goal of the research was to develop the best possible algorithms for detection of drowsiness, based on measures that could be computed on-board in a vehicle. Additional objectives were that developed

algorithms would produce low false alarm rates, that there should be minimal encumbering of (interference with) the driver, and that the algorithms should be-suitable for later field testing.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Currently various technologies are used to detect driver drowsiness some of them are: Steering pattern monitoring which primarily uses steering input from electric power steering system, vehicle position in lane monitoring which uses lane monitoring camera and physiological measurement which requires body sensors to measure parameters like brain activity, heart rate, skin conductance, muscle activity.

3.1.1 Disadvantage of Existing System

- The driver always needs to have a constant contact with the existing system technologies.
- Requires large investments to implement the existing technologies.
- Existing technologies are very bulky.

3.2 PROPOSED SYSTEM

Camera mounted on the dash board of a car to capture the images of the driver for video acquisition. Raspberry Pi is a credit card sized single-board computer for implementation and an audio alarm system for drowsiness detection and correction. In the proposed system, first video acquisition is achieved by camera placed in front of the driver. The acquired video is then converted into a series of frames/images. The next step is to detect the driver's face, in each and every frame extracted from the video. Next, we detect the eyes by processing only the region of interest. Once the eyes have been detected, the next step is to determine whether the eyes are in open/closed state. Monitoring each frame for pupil detection if the eyes are detected to be open, no action is taken. But if eyes are detected to be closed continuously and

it is above threshold value then it means that the automobile driver is feeling drowsy and a sound alarm is triggered.

3.2.1 Advantages of Proposed System

1. Reduction in road accident.
2. No physical connection required with driver.
3. Easy to install and repair
4. Low cost and secure

CHAPTER 4

SYSTEM REQUIREMENT SPECIFICATION

4.1 Software and Hardware Requirements

Software and hardware requirements for the application are derived from industry standard technologies and widely available commercial products. Based on specific configuration, there may be other optional components. The minimum requirements are described in the sections which follow.

4.1.1 Hardware Requirements

- Raspberry Pi 4 Model-B with 2GB RAM.
- High Quality Camera.
- Alarm System.
- Power Supply.

4.1.2 Software Requirements

- Python.
- OpenCV.
- Linux.

CHAPTER 5

SYSTEM DESIGN

5.1 System Architecture

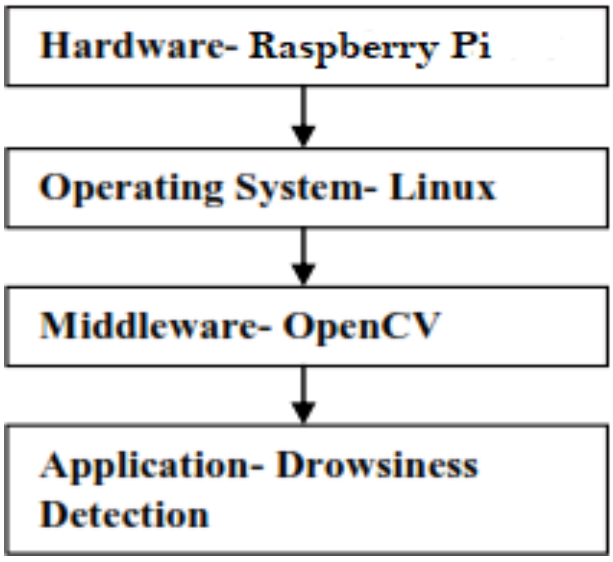

Fig 5.1 System Architecture

This project delivers a drowsiness detection mechanism along the following lines. As indicated, the whole system can be delivered in an ARM environment or an Intel environment. The processor is left to the choice of the users' feasibility and viability. The system proposed is built on Linux Operating system and the detection mechanism is carried out with the help of OpenCV Library. Linux is an interface between computer/server hardware, and the programs which run on it. The most obvious advantage of using Linux is that it is free to obtain. A Linux distribution can be installed on any number of computers free of cost. In line with the costs, the security aspect of Linux is much stronger than that of other OS. Hence, no extra money has to be spent on virus protection software. Open source Computer Vision Library (OpenCV) is a library of programming functions that is exclusively used for applications based on computer vision. It has C++, C and Java interfaces and supports Linux easily. OpenCV is quick when it comes to speed of execution. OpenCV programs only require around 70MB of RAM to

run in real- time. Any device that can run C, can, in all probability, run OpenCV. The detection algorithm is implemented using features of the Haar Classifiers for object detection. The core basis for Haar classifier object detection is the Haar-like features. These features use the change in contrast values between adjacent rectangular groups of pixels instead of the intensity values of a pixel. The entire system for detecting drowsiness is implemented using these Haar Classifiers.

The different phases of the algorithm are driven by:

- Face detection
- Eye detection
- Drowsiness detection

5.2 Flow Chart

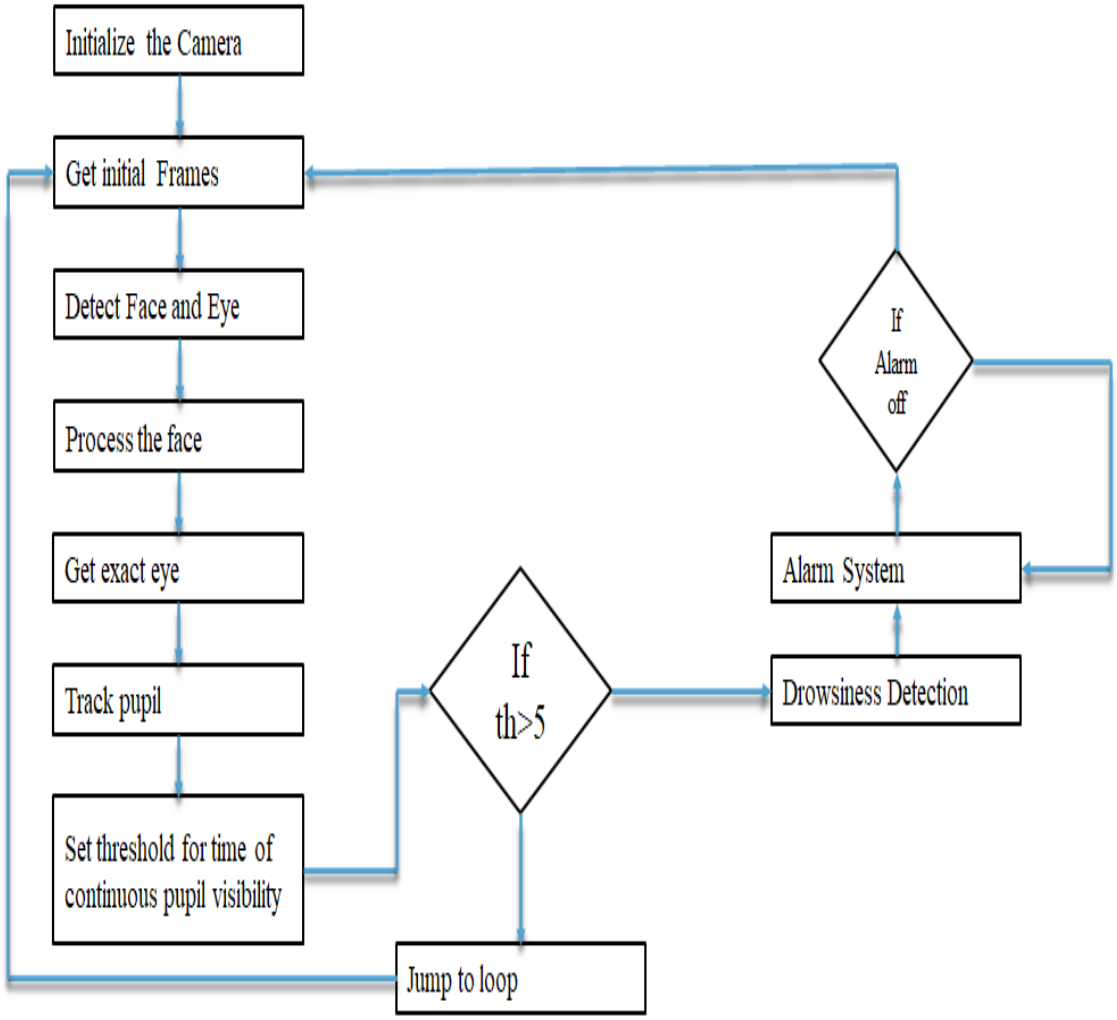

Fig 5.2 Flow Chart

5.3 Use Case Diagram

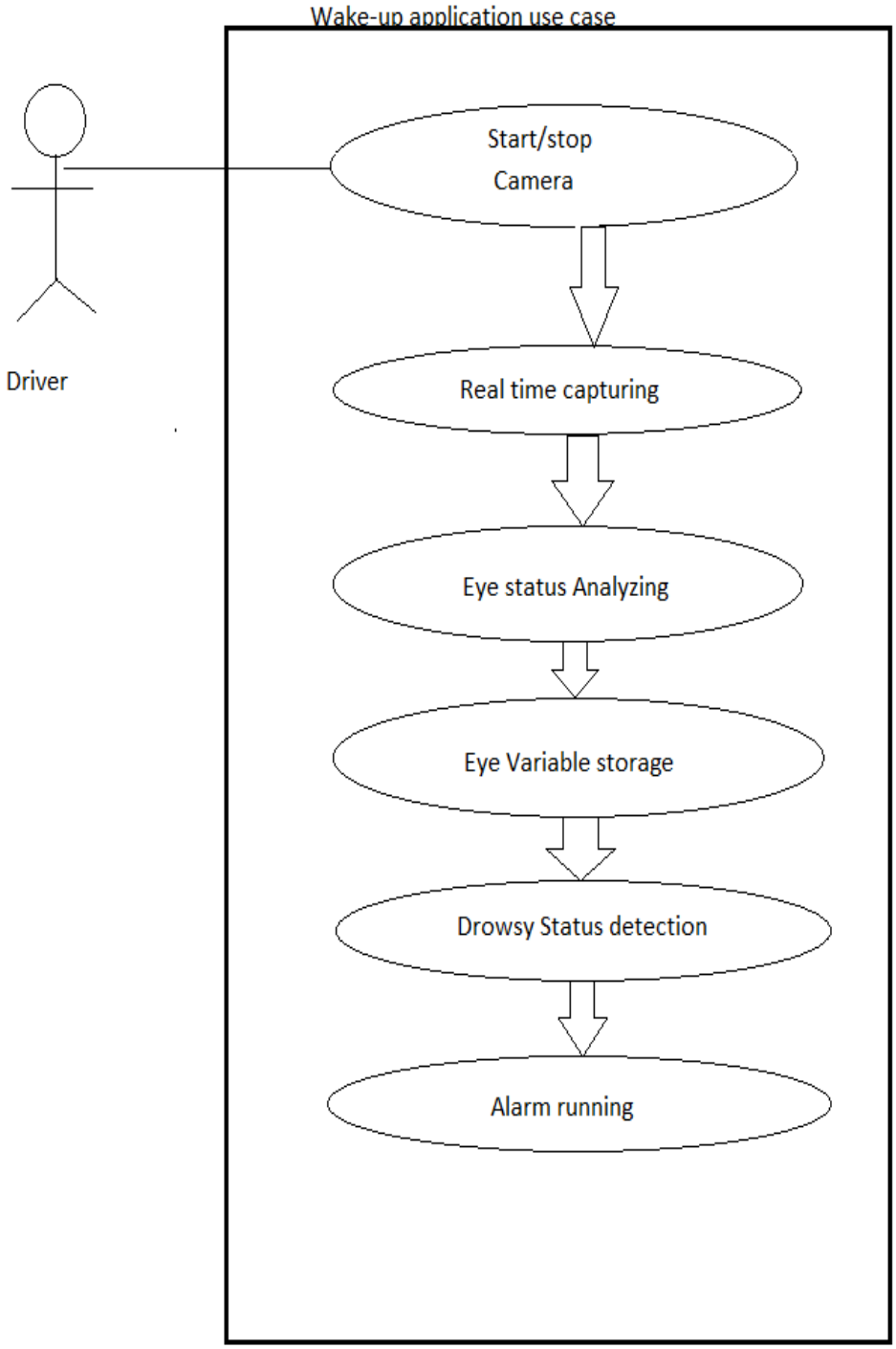

Fig 5.3 Use Case

5.4 Sequence Diagram

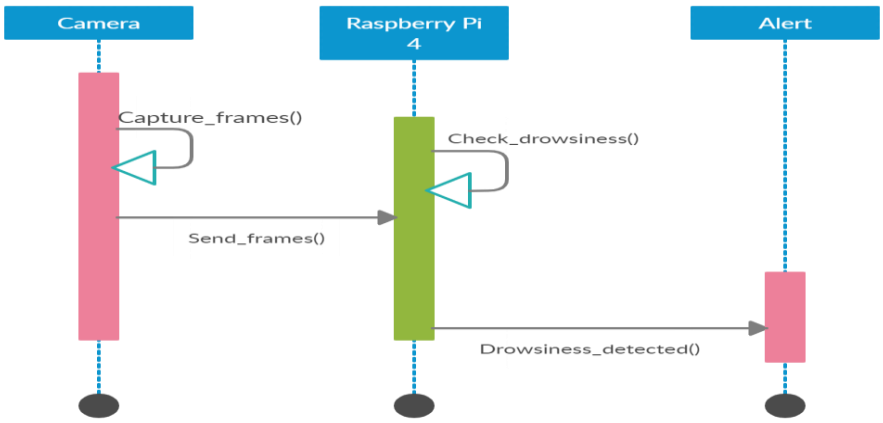

Fig 5.4 Sequence Diagram

CHAPTER 6

IMPLEMENTATION

6.1 Introduction

Implementation is the stage in the project where the theoretical design is turned into a working system and is giving confidence on the new system for the users, which it will work efficiently and effectively. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the change-over, an evaluation, of change over methods. Apart from planning major task of preparing the implementation are education and training of users. The more complex system being implemented, the more involved will be the system analysis and the design effort required just for implementation. An implementation co-ordination committee based on policies of individual organization has been appointed. The implementation process begins with pre-paring a plan for the implementation of the system. According to this plan, the activities are to be carried out, discussions made regarding the equipment and resources and the additional equipment has to be acquired to implement the new system.

Implementation is the final and important phase, the most critical stage in achiev-ing a successful new system and in giving the users confidence. That the new system will work is effective .The system can be implemented only after through testing is done and if it found to working according to the specification. This method also offers the greatest security since the old system can take over if the errors are found or inability to handle certain type of transactions while using the new system.

User Training

After the system is implemented successfully, training of the user is one of the most important subtasks of the developer. For this purpose user manuals are prepared and han-dled over to the user to operate the developed system. Thus the users are trained to operate the developed systems successfully in future.

6.2 Code Used

```
# 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
from imutils.video import VideoStream
from imutils import face_utils
import numpy as np
import argparse
import imutils
import time
import dlib
from threading import Thread
import cv2
import RPi.GPIO as GPIO
from time import sleep
from subprocess import call


GPIO.setmode(GPIO.BOARD)
GPIO.setup(03, GPIO.OUT)
pwm=GPIO.PWM(03, 50)
def SetAngle(angle):
    duty = angle / 18 + 2
    GPIO.output(03, True)
    pwm.ChangeDutyCycle(duty)
    sleep(1)
```

```
        GPIO.output(03, False)
        pwm.ChangeDutyCycle(0)

def setMotor(angle):
            pwm.start(0)
            if angle==1:
                    SetAngle(47)
            elif angle==-1:
                    SetAngle(42)

            pwm.stop()

def rotateDegree(boxX, boxWidth, width = 450):
    # calculate middle
    #  difference = middle of view - middle of box

    difference = (width/float(2)) - (boxX + boxWidth/float(2))
    # print(difference)
    if (-10 < difference < 10):
        return 0

    return np.sign(difference)
    # if difference > 0:
    #     return -1 # L
    # elif difference < 0:
    #     return 1 # R
    # else:
    #     return 0 # 0

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 Alarm-espeak 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...")

cmd_beg= 'espeak '
cmd_end= ' 2>/dev/null'
# 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 = 0.3
EYE_AR_CONSEC_FRAMES = 16

# 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...")
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)
pwm.start(0)
SetAngle(45)
pwm.stop()
# loop over frames from the video stream
while True:
        # grab the frame from the threaded video file stream, resize
        # it, and convert it to grayscale
        # channels)
        frame = vs.read()
        frame = imutils.resize(frame, width=450)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # 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)
mouth_middle = shape[67]
face_end = shape[14]

degree = rotateDegree(x, w)
print(degree)
setMotor(degree)

# 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

                        # check to see if the TrafficHat buzzer should
                        # be sounded
                        if args["alarm"] > 0:
                            call([cmd_beg+'Wake up you are dozing off
again' +cmd_end], shell=True)
                        #       th.buzzer.blink(0.1, 0.1, 10,
                        #               background=True)
                        #if args["alarm"] != "":
                        #       t = Thread(target=sound_alarm,
                        #               args=(args["alarm"],))
                        #       t.deamon = True
                        #       t.start()

                    # draw an alarm on the frame
                    cv2.putText(frame, "DROWSINESS ALERT!", (10, 30),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0,
                            255), 2)

            # otherwise, the eye aspect ratio is not below the blink
            # threshold, so reset the counter and alarm
            else:
                COUNTER = 0
                ALARM_ON = False
```

```
                # 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)
        key = cv2.waitKey(1) & 0xFF

        # if the `q` key was pressed, break from the loop
        if key == ord("q"):
                break
GPIO.cleanup()
# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```

CHAPTER 7

SYSTEM TESTING AND RESULTS

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

7.1 Test strategy and approach

Testing is performed manually and functional tests are written in detail.

7.2 Features to be tested

- To verify that all the units are working correctly.
- No duplicates user entries should be allowed.
- Network signal must be available.

7.3 Test Results

- All the test cases mentioned above passed successfully. No defects encountered.

7.4 Test case 1: When there is ambient light

Fig 7.4. Test Scenario #1 – Ambient lighting

Result: As shown in Fig 7.4, when there is ambient amount of light, the automobile driver's face and eyes are successfully detected.

7.5 Test case 2: Position of the automobile drivers face

7.5.1 Center Positioned

Fig 7.5.1. Test Scenario Sample#2 driver face in center of frame

RESULT: As shown in Fig 4, When the automobile driver's face is positioned at the Centre, the face, eyes, eye blinks, and drowsiness was successfully detected.

7.5.2 Right Positioned

Fig 7.5.2. Test Scenario Sample#3 – driver face to right of frame

RESULT: As shown in Fig 7.5.2, When the automobile driver's face is positioned at the Right, the face, eyes, eye blinks, and drowsiness was successfully detected.

7.5.3 Left Positioned

Fig 7.5.3. Test Scenario Sample#4- driver face to left of frame

RESULT: As shown in screen snapshot in Fig 6,when the automobile driver's face is positioned at the Left, the face, eyes , eye blinks, and drowsiness was successfully detected.

7.6 Test case 3: When the automobile driver is wearing spectacles

Fig 7.6. Test Scenario Sample#5 – Driver with spectacles

RESULT: As shown in screen snapshot in Fig 7.6, When the automobile driver is wearing spectacles, the face, eyes , eye blinks, and drowsiness was successfully detected.

7.7 Test case 4: When the automobile driver's head's tilted

Fig 7.7. Test Scenario Sample#6 – Various posture

RESULT: As shown in screen snapshot in Fig 8,when the automobile driver's face is tilted for more than 30 degrees from vertical plane, it was observed that the detection of face and eyes failed.

The system was extensively tested even in real world scenarios, this was achieved by placing the camera on the visor of the car , focusing on the automobile driver. It was found that the system gave positive output unless there was any direct light falling on the camera.

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

Conclusion

Driver drowsiness detection is designed mainly to keep the driver awake while driving to avoid the accident due to sleepiness. The alert signal is generated from embedded device to wake up the driver from sleepy state. The Raspberry Pi along with camera is used to calculate the drowsiness of the driver in real time. Fatigue is measured by detecting Eye and face using Haar Cascade Classifier, especially facial landmarks is detected using shape-predictor and Eye Aspect Ratio (EAR) by calculating the Euclidean distance between the eyes. Accurate eye detection and faces in every frame will help to calculate drowsiness level. Frequent detection of eye blinkingis measured properly and it helps to indicates drowsiness. When he/she reaches maximum threshold the driver will be alarmed by a loud warning that will wake up the driver from the sleep state.

Future scope

The system at this stage is a "Proof of Concept" for a much substantial endeavor. This will serve as a first step towards a distinguished technology that can bring about an evolution aimed at ace development. The developed system has special emphasis on real-time monitoring with flexibility, adaptability and enhancements as the foremost requirements.

Future enhancements are always meant to be items that require more planning, budget and staffing to have them implemented. There following are couple of recommended areas for future enhancements:

- Standalone product: It can be implemented as a standalone product, which can be installed in an automobile for monitoring the automobile driver.
- Smart phone application: It can be implemented as a smart phone application, which can be installed on smart phones. And the automobile driver can start the application after placing it at a position where the camera is focused on the driver.

APPENDIX A

1. Raspberry Pi 4

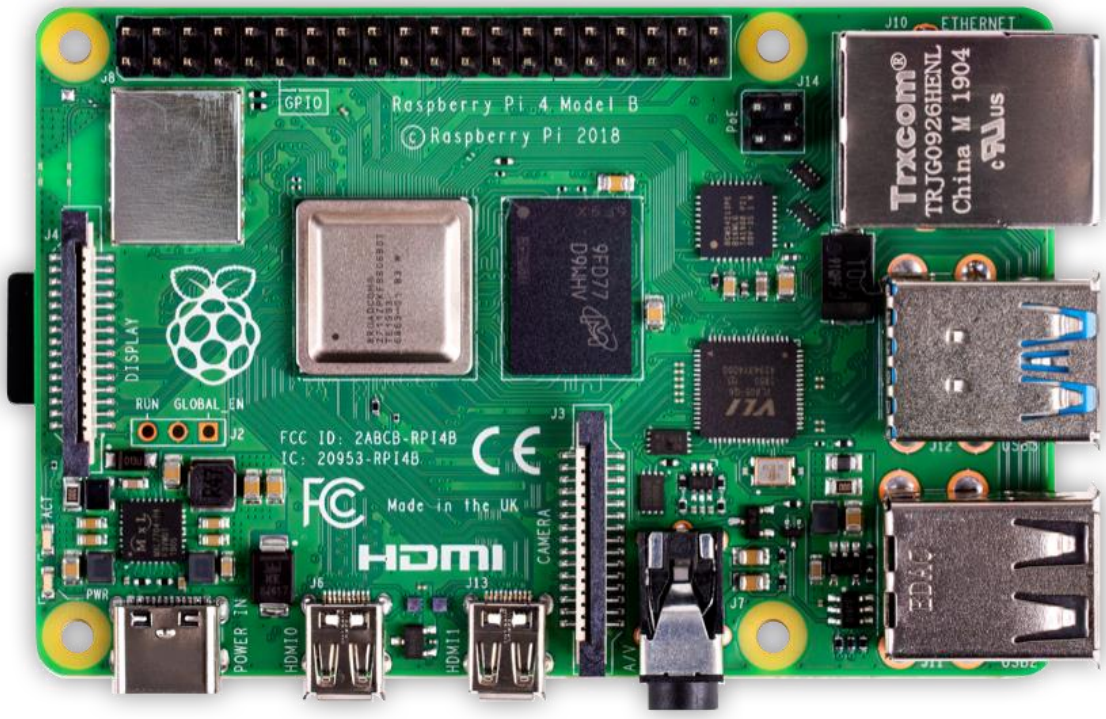

Fig 6.1.1 Raspberry Pi 4

❖ Raspberry Pi 4 Specifications:

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- 2GB LPDDR4-3200 SDRAM (depending on model)
- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 USB 3.0 ports; 2 USB 2.0 ports.
- Raspberry Pi standard 40 pin GPIO header (fully backwards compatible with previous boards)
- 2 × micro-HDMI ports (up to 4kp60 supported)
- 2-lane MIPI DSI display port

- 2-lane MIPI CSI camera port
- 4-pole stereo audio and composite video port
- H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode)
- OpenGL ES 3.0 graphics
- Micro-SD card slot for loading operating system and data storage
- 5V DC via USB-C connector (minimum 3A*)
- 5V DC via GPIO header (minimum 3A*)
- Operating temperature: 0 – 50 degrees C ambient

2. USB Camera

Fig 6.1.2 USB Camera

❖ Camera Specifications:

- Still resolution: 5 Megapixels
- Video mode: 720p30fps
- Video format: h.264
- Horizontal field of view: 62 degrees
- Vertical field of view: 49 degrees

3. Alarm

Fig 6.1.3 Alarm

❖ Alarm Specifications:

- Module driven by 9012 triode
- Working voltage of 3.3 V to 5 V
- Sets of fixed bolt hole, convenient installation
- Small board PCB size: 3.3 cm X 1.3 cm
- Raspberry Pi Compatible

APPENDIX B

1. LINUX

Linux is a family of open source Unix-like operating systems based on the Linux kernel, an operating system kernel first released on September 17, 1991, by Linus Torvalds. Linux is typically packaged in a Linux distribution. Distributions include the Linux kernel and supporting system software and libraries, many of which are provided by the GNU Project. Many Linux distributions use the word "Linux" in their name, but the Free Software Foundation uses the name GNU/Linux to emphasize the importance of GNU software, causing some controversy. Popular Linux distributions include Debian, Fedora, and Ubuntu. Commercial distributions include Red Hat Enterprise Linux and SUSE Linux Enterprise Server. Desktop Linux distributions include a windowing system such as X11 or Wayland, and a desktop environment such as GNOME or KDE Plasma. Distributions intended for servers may omit graphics altogether, or include a solution stack such as LAMP. Because Linux is freely redistributable, anyone may create a distribution for any purpose. Linux was originally developed for personal computers based on the Intel x86 architecture, but has since been ported to more platforms than any other operating system. Because of the dominance of Android on smartphones, Linux also has the largest installed base of all general-purpose operating systems. Although it is used by only around 2.3 percent of desktop computers, the Chromebook, which runs the Linux kernel-based Chrome OS. Linux is the leading operating system on servers (over 96.4% of the top 1 million web servers' operating systems are Linux), leads other big iron system such as mainframe computers, and is the only OS used on TOP500 supercomputers. Linux also runs on embedded systems, i.e. devices whose operating system is typically built into the firmware and is highly tailored to the system. Linux is one of the most prominent examples of free and open-source software collaboration. The source code may be used, modified and distributed—commercially or non-commercially—by anyone under the terms of its respective licenses, such as the GNU General Public License.

2. PYTHON

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL), capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's Benevolent Dictator For Life, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. He now shares his leadership as a member of a five-person steering council. In January 2019, active Python core developers elected Brett Cannon, Nick Coghlan, Barry Warsaw, Carol Willing and Van Rossum to a five-member "Steering Council" to lead the project. Python 2.0 was released on 16 October 2000 with many major new features, including a cycle-detecting garbage collector and support for Unicode. Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely backward-compatible. Many of its major features were backported to Python 2.6.x and 2.7.x version series. Releases of Python 3 include the 2to3 utility, which automates (at least partially) the translation of Python 2 code to Python 3. Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3.

3. OPEN CV

Officially launched in 1999 the OpenCV project was initially an Intel Research initiative to advance CPU-intensive applications, part of a series of projects including real-time ray tracing and 3D display walls. The main contributors to the project included a number of optimization experts in Intel Russia, as well as Intel's Performance Library Team. In the early days of OpenCV, the goals of the project were described as:

- Advance vision research by providing not only open but also optimized code f or basic vision infrastructure. No more reinventing the wheel.

- Disseminate vision knowledge by providing a common infrastructure that developers could build on, so that code would be more readily readable and transferable.
- Advance vision-based commercial applications by making portable, performance-optimized code available for free – with a license that did not require code to be open or free itself.

The first alpha version of OpenCV was released to the public at the IEEE Conference on Computer Vision and Pattern Recognition in 2000, and five betas were released between 2001 and 2005. The first 1.0 version was released in 2006. The second major release of the OpenCV was in October 2009. OpenCV 2 includes major changes to the C++ interface, aiming at easier, more type-safe patterns, new functions, and better implementations for existing ones in terms of performance (especially on multi-core systems). Official releases now occur every six months and development is now done by an independent Russian team supported by commercial corporations. In August 2012, support for OpenCV was taken over by a non-profit foundation OpenCV.org, which maintains a developer and user site. On May 2016, Intel signed an agreement to acquire Itseez, a leading developer of OpenCV.

BIBLIOGRAPHY

1. P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

2. W. Dong and X. Wu, "Driver fatigue detection based on the distance of eyelid," in *Proceedings of the 2005 IEEE International Workshop on VLSI Design and Video Technology, IWVDVT 2005*, IEEE, pp.365–368, China, May 2005.

3. A. Dasgupta, A. George, S. L. Happy, and A. Routray, "A vision-based system for monitoring the loss of attention in automotive drivers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1825–1838, 2013.

4. N. Alioua, A. Amine, M. Rziza, and D. Aboutajdine, "Eye state analysis using iris detection based on Circular Hough Transform," in *Proceedings of the 2011 International Conference on Multimedia Computing and Systems, ICMCS'11*, IEEE, pp. 1–5, Morocco, April 2011.

5. Y. Du, P. Ma, X. Su, and Y. Zhang, "Driver fatigue detection based on eye state analysis," in *Proceedings of the 11th Joint Conference on Information Sciences*, 2008.X. Liu, X. Tan, and S. Chen, "Eyes closeness detection using appearance based methods," in *Proceedings of the International Conference on Intelligent Information Processing*, vol. 385, pp. 398–408, Springer, 2012.

6. G. Fa-deng and H. Min-xian, "Study on the detection of locomotive driver fatigue based on image," in *Proceedings of the 2010 2nd International Conference on Computer Engineering and Technology*, pp. V7-612–V7-615, Chengdu, China, April 2010.

7. L. Zhou and H. Wang, "Open/closed eye recognition by local binary increasing intensity patterns," in *Proceedings of the 2011 IEEE 5th International Conference on Robotics, Automation and Mechatronics,RAM 2011*, IEEE, pp. 7–11, China, September 2011.M. Tafreshi and A. M. Fotouhi, "A fast and accurate algorithm for eye opening or closing detection based on local maximum vertical derivative pattern," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 24, no. 6, pp. 5124–5134, 2016.

8. H. Qin, J. Liu, and T. Hong, "An eye state identification method based on the embedded hidden Markov model," in *Proceedings of the 2012 IEEE International Conference on Vehicular Electronics and Safety, ICVES 2012*, IEEE, pp. 255–260, Turkey, July 2012.