MANIPAL ACADEMY of HIGHER EDUCATION
(Deemed to be University under Section 3 of the UGC Act, 1956)

MANIPAL SCHOOL OF INFORMATION SCIENCES
(A Constituent unit of MAHE, Manipal)

DROWSY DRIVER DETECTION USING MACHINE LEARNING

Reg. Number	Name	Branch
221039002	VISHAK V K	EMBEDDED SYSTEMS
221039004	SANDESH B S	EMBEDDED SYSTEMS

Under the guidance of

Dr. Mohan Kumar J
Faculty
Manipal School of Information Sciences,MAHE, MANIPAL

DECEMBER 2022

MANIPAL SCHOOL OF INFORMATION SCIENCES
MANIPAL
(A constituent unit of MAHE, Manipal)

DECLARATION

We hereby declare that the term project work entitled “Drowsy driver detection using machine learning” has been carried out with **Dr.Mohan Kumar J**. This work is an original or emulated work, which was carried out for learning as a term project for 1st Semester ME(Embedded Systems).

Vishak V K

Sandesh B S

Place: Manipal

Date: 22/12/2022

MANIPAL SCHOOL OF INFORMATION SCIENCES
MANIPAL
(A constituent unit of MAHE, Manipal)

Declaration of originality of work and report by the students

We have not committed plagiarism in any of the form described in the 'MAHE Plagiarism policy'. We have documented all methods, data, and processes truthfully and we have not manipulated any data. We have mentioned all the persons who are significant facilitators of the work. The work has been screened electronically for plagiarism. References, help and material obtained from other sources have been duly acknowledged.

Vishak V K	221039002
Sandesh B S	221039004

Place: Manipal Date: 22/12/2022

MANIPAL SCHOOL OF INFORMATION SCIENCES
MANIPAL
(A constituent unit of MAHE, Manipal)

CERTIFICATE

This is to certify that this thesis report entitled **Drowsy driver detection using machine learning** is a bonafide project work carried out and completed by us, **Vishak V K** and **Sandesh B S** . This is an original piece /emulated work, done as per the regulations of the **Manipal Academyof Higher Education, Manipal** for the purpose of project learning.

ACKNOWLEDGEMENT

We wish to express our sincere gratitude to everyone who helped and guided us in completingthis project work.

We would also like to express our sincere gratitude to our project guide, Mohan Kumar J, forhis invaluable advice, guidance and feedback for the project.

We are also thankful to all the panel members for their assistance and encouragement.

ABSTRACT

Drowsy driving is a major problem in the World. Drowsy driving is the dangerous combination of driving and sleepiness or fatigue. This usually happens when a driver has not slept enough, but it can also happen due to untreated sleep disorders, medications, drinking alcohol, or shift work. This project is mainly based on Haar-Cascade Classifier which is the algorithm to detect drowsy eyes by using XML (Extensible Markup Language) files of Haar cascade in OpenCV-Python we can detect single eye movement if the eye is closed for more than the normal eye blink rate then this algorithm will detect the drowsiness and it will signal to the buzzer to beep and at the same time slow down the motor. The complete system is implemented on Raspberry Pi 3 Model B+ which uses a webcam to monitor user's eye blink rate and average blink duration to detect drowsiness. The project is designed for vehicle safety which helps preventing accidents caused by the driver getting drowsy.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 INTRODUCTION

1.1 Drowsy driving and causes

Drowsy driving is one of the major causes behind fatal road accidents. One of the recent study 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 2015, 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. All this led to the development of Intelligent Driver Assistance Systems. Real time drowsy driving detection is one of the best possible major 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 paper, we are presenting technique to detect driver drowsiness using of Open CV, raspberry pi and image processing. Image Processing means processing digital image by means of a computer. This performs some mathematical functions and operations on images or on videos. The output that we get after image processing is a set of parameters or some altered image, images or videos. Due to the low price of the Raspberry Pi, it is being used for image processing and video processing in many projects.

This project is mainly focused on haar cascade classifiers. Haar-like features, developed by Viola and Jones. A Haar-like feature considers neighboring rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums. This difference is then used to categorize subsections of an image. An example of this would be the detection of human faces. Commonly, the areas around the eyes are darker than the areas on the cheeks. One example of a Haar-like feature for face detection is therefore a set of two neighboring rectangular areas above the eye and cheek regions. The cascade classifier consists of a list of stages, where each stage consists of a list of weak learners. The system detects objects in question by moving a window over the image. Each stage of the classifier labels the specific region defined by the current location of the window as either positive or negative. Positive meaning that an object was found or negative means that the specified object was not found in the image. If the labelling yields a negative result, then the classification of this specific region is hereby complete and the location of the window is moved to the next location. If the labelling gives a positive result, then the region moves of to the next stage of classification. The classifier yields a final verdict of positive, when all the stages, including the last one, yield a result, saying that the object is found in the image. A true positive means that the object in question is indeed in the image and the classifier labels it as such a positive result. A false positive means that the labelling process falsely determines, that the object is located in the image, although it is not. A false negative occurs when the classifier is unable to detect the actual object from the image and a true negative means that a non-object was correctly classifier as not being the object in question. In order to work well, each stage of the cascade must have a low false negative rate, because if the actual object is classified as a non-object, then the classification of that branch stops, with no way to correct the mistake made. However, each stage can have a relatively high false positive rate, because even if the n-th stage classifies the non-object as actually being the object, then this mistake can be fixed in n+1-th and subsequent stages of the classifier.

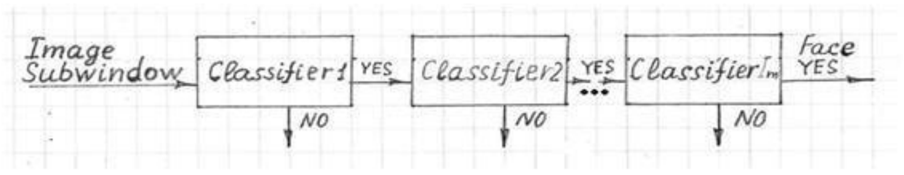


Figure 1.1: Stages of the cascade classifier

So by using the principle of haar cascade XML (Extensible Markup Language) files are developed which detect eyes and faces. Using this XML files in OpenCV-Python we can detect the eye movement.

1.2 Image processing

Image processing is a method to perform some operations on an image, in order to extract some useful information from it. An image is nothing more than a two dimensional matrix (3-D in case of colored images) which is defined by the mathematical function f(x,y) where x and y are the two co-ordinates horizontally and vertically. The value of f(x,y) at any point is gives the pixel value at that point of an image, the pixel value describes how bright that pixel is, and/or what color it should be.For grayscale images the pixel value is a single number that represents the brightness of that pixel, the most common pixel format is the byte image, which is stored as an 8-bit integer giving a range of possible values from 0 to 255. As a convention is taken to be black, and 255 is taken to be white the values in between make up the different shades of gray. To represent color images, separate red, green and blue components must be specified for each pixel (assuming a RGB color model), and so the pixel `value’ becomes a vector of three numbers. Often the three different components are stored as three separate `grayscale’ images known as color planes (one for each of red, green and blue), which have to be recombined when displaying or processing. In haar cascade the colored image is converted into grayscale image then the algorithms is applied to detection of the object.

1.3 Different Methods of image processing

1.3.1 Haar cascade classifier

It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, Haar features shown in the below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under the white rectangle from sum of pixels under the black rectangle.

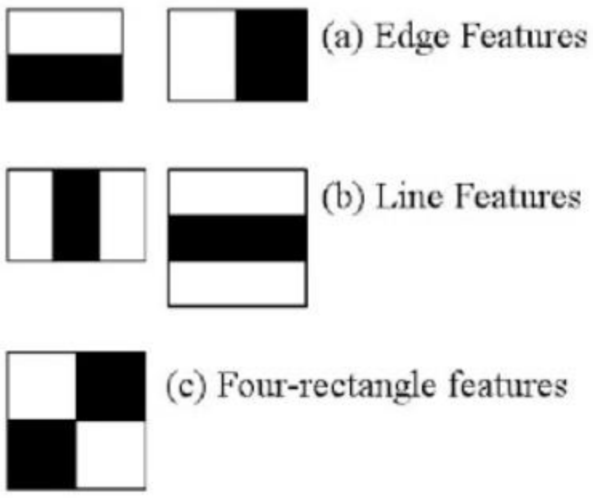


Figure 1.2: Haar features

Now, all possible sizes and locations of each kernel are used to calculate lots of features. For each feature calculation, we need to find the sum of the pixels under white and black rectangles. To solve this, they introduced the integral image. However large your image, it reduces the calculations for a given pixel to an operation involving just four pixels.It makes things super-fast.

1.3.2. Histogram of Oriented Gradients (HOG) in Dlib

The second most popular implement for face detection is offered by Dlib and uses a concept called Histogram of Oriented Gradients (HOG). The idea behind HOG is to extract features into a vector, and feed it into a classification algorithm like a Support Vector Machine for example that will assess whether a face (or any object you train it to recognize actually) is present in a region or not.The features extracted are the distribution (histograms) of directions of gradients (oriented gradients) of the image. Gradients are typically large around edges and corners and allow us to detect those regions.

Chapter 2 LITERATURE REVIEW

Paper 1

Puja Seemar, Anurag Chandna proposed *DROWSY DRIVER DETECTION USING IMAGE PROCESSING. INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY* on July, 2017.

The major driver errors are caused by drowsiness, drunken and reckless behavior of the driver. This paper focuses on a driver drowsiness detection system in Intelligent Transportation System, which focuses on abnormal behavior exhibited by the driver using Raspberry pi single board computer. The capability of driving support systems to detect the level of driver's alertness is very important in ensuring road safety. By observation of blink pattern and eye movements, driver fatigue can be detected early enough to prevent collisions caused by drowsiness. In the proposed system a nonintrusive driver drowsiness monitoring system has been developed using computer vision techniques. Based on the simulation results, it was found that the system has been able to detect drowsiness in spite of driver wearing spectacles as well as the darkness level inside the vehicle. Moreover the system is capable of detecting drowsiness within time duration of about two seconds. The detected abnormal behavior is corrected through alarms in real time. They have also done a transformation of an image from 2d to 3d using wavelet analysis.

Here they also compared the wavelet technique with other techniques namely stereo-photogrammetric & edge information technique. The image conversion from 2d to 3d can be done by finding the edges of an image. This edge detection can be used in various fields of image processing, image analysis, image pattern recognition, and computer vision, as well as in human vision. In this, we did our experiment using wavelet technique and the results when compared with the stereo photogrammetry and edge information techniques they found that the wavelet technique gives better result.

Paper 2

Dwipjoy Sarkar, Atanu Chowdhury proposed *A Real Time Embedded System Application for Driver Drowsiness and Alcoholic Intoxication Detection*. Volume 10 Number 9 - Apr 2014.

This paper outlines a novel approach for the real time detection of car driver drowsiness and alcoholic intoxication. There are large numbers of road accidents which takes place due to fatigue or alcohol drinking of driver. Computer vision and alcohol gas sensor application is combined to an embedded system to achieve this goal. The proposed system is realized with an open source 5 megapixel digital camera supported embedded system board Raspberry-pi loaded with Raspbian-OS, and Python-IDLE with Open-CV installed. The Raspberry-pi system board is serially interfaced with another open source embedded system board Arduino Uno with I2C protocol, which will perform some task like issuing the alarm notification and switching off the car power source to stop the car upon receiving the positive detection message from Raspberry-pi.

Paper 3

Tejasweeni Musale,B. H. Pansambal proposed *Real Time Driver Drowsiness Detection system using Image processing*.International Journal for Research in Engineering Application & Management Vol-02, Issue 08, Nov 2016.

One of the major cause behind the road accidents is driver's drowsiness. Thus, countermeasure device is currently essential in many fields for sleepiness related accident prevention. Real-time driver drowsiness system alerts users when they are falling asleep. The project is designed to combat narcolepsy and microsleep. Microsleep strikes quickly. Users probably don't even realize that they are in the process of falling asleep, and almost certainly don't notice that eye blinking for longer than usual. The implemented project is mainly based on three components 1) Face and Eye detection: Performs scale invariant detection using Haar Cascade Classifier perform through a webcam. 2) Eye feature extraction: Eye features are extracted using Hough Circle and 3) Extract single eye and perform drowsiness detection on it. Whereas the complete system is implemented on Raspberry Pi which uses a webcam to monitor user's eye blink rate and average blink duration to detect drowsiness. The project is designed for a car safety which helps prevent accidents caused by the driver getting drowsy.

Chapter 3 OBJECTIVE OF THE PROJECT

3.1 Objective

1. To learn the drowsy driver image concepts and OpenCV.
2. To implement an algorithm for drowsy driver detection using OpenCV.
3. To implement a machine learning model to identify the drowsy drive

Chapter 4 PROPOSED SYSTEM/SOFTWARE BLOCK DIAGRAM

4.1 Proposed Block Diagram

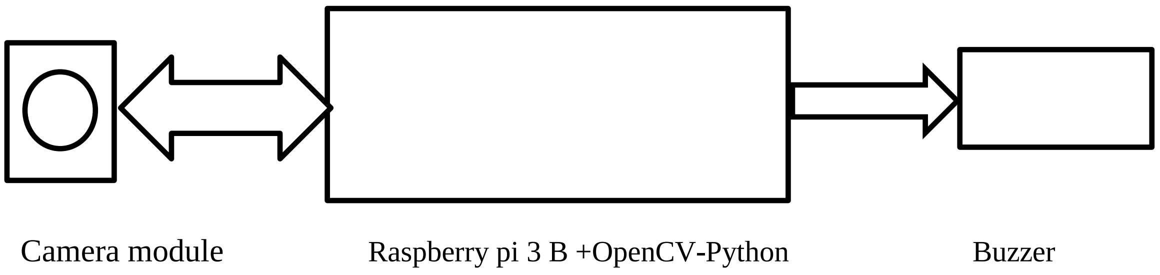


Figure 4.1: Proposed Block Diagram

4.2 Flow chart

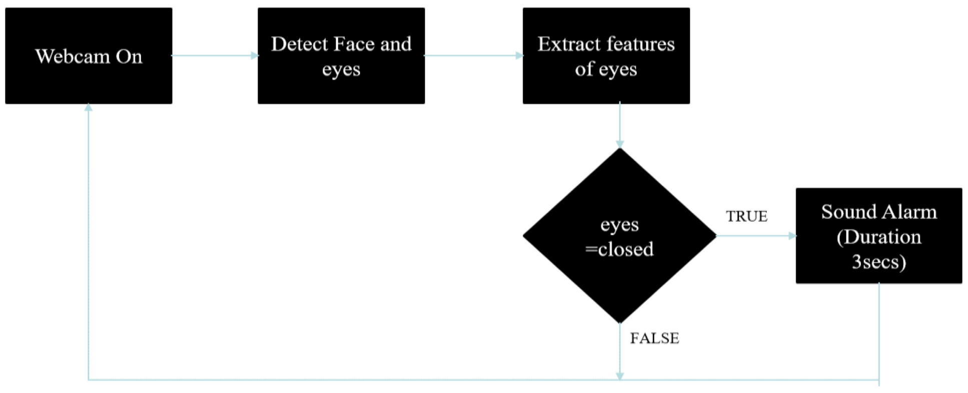


Figure 4.2: Flow Chart representing flow of camera and image detection

1. The webcam will be kept in a while loop, by keeping it turned on in all the conditions except when character ‘Q’ is input.

2. The next step is for the webcam to detect the face and eyes of the driver.

3. Once the face detection is completed, we will proceed ahead with the feature extraction of the eyes.

4. This feature extraction is done to determine whether the eyes are closed or opened.

5. If the eyes are closed, an alarm will be rung for a time duration of 3 secs to alert the driver and an alert ‘Warning eyes are closed’ is displayed.

6. There will be no changes in the alarm status if the eyes are open.

Chapter 5 FINAL BLOCK DIAGRAM

The proposed system consists of three components 1. Capturing: Camera mounted on the automotive dashboard captures the images of drivers face including eyes.2. Processing and Detecting: Captured facial image is used to determine drivers eye i.e. open or closed. The driver’s current eye state can be determined using different types of HARR classifier cascades XML files.3. Signaling: In case of abnormal behavior that is drivers eyes found to be closed as a corrective action alarm signal will be raised. The Raspberry pi, a single board computer which is connected serially to the PC is responsible for the corrective necessary actions.

The proposed system is shown in Figure 7.1. The primary intent is to detect driver drowsy condition with real time by quickly processing the input data. The number of frames in which the eyes are found closed is monitored and counted. If the number of frames exceeds a threshold value, then a warning is generated showing that the drowsiness is detected. All these criteria have been well satisfied by choosing the system with the appropriate classifiers in OpenCV for eye closure detection. In this algorithm, first a driver’s image is captured by web camera for processing. In OpenCV, the drivers face detection from the captured image is carried out first, and then followed by the eye detection. 'haarcascade_eye_tree_eyeglasses.xml' and 'haarcascade_frontalface_alt.xml' these two XML files have been used to locate eye position on the face. With the eye detection technique we will only be able to detect the open state of eyes. The algorithm then counts the number of open eyes form each frame and determines the drowsiness. If the criteria are satisfied, then the driver is said to be drowsy. The buzzer connected to the system performs actions to correct the driver abnormal behavior.

5.1 Design and implementation

Eye and face extraction from frame:

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, “Rapid Object Detection using a Boosted Cascade of Simple Features” in 2001.

It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier.

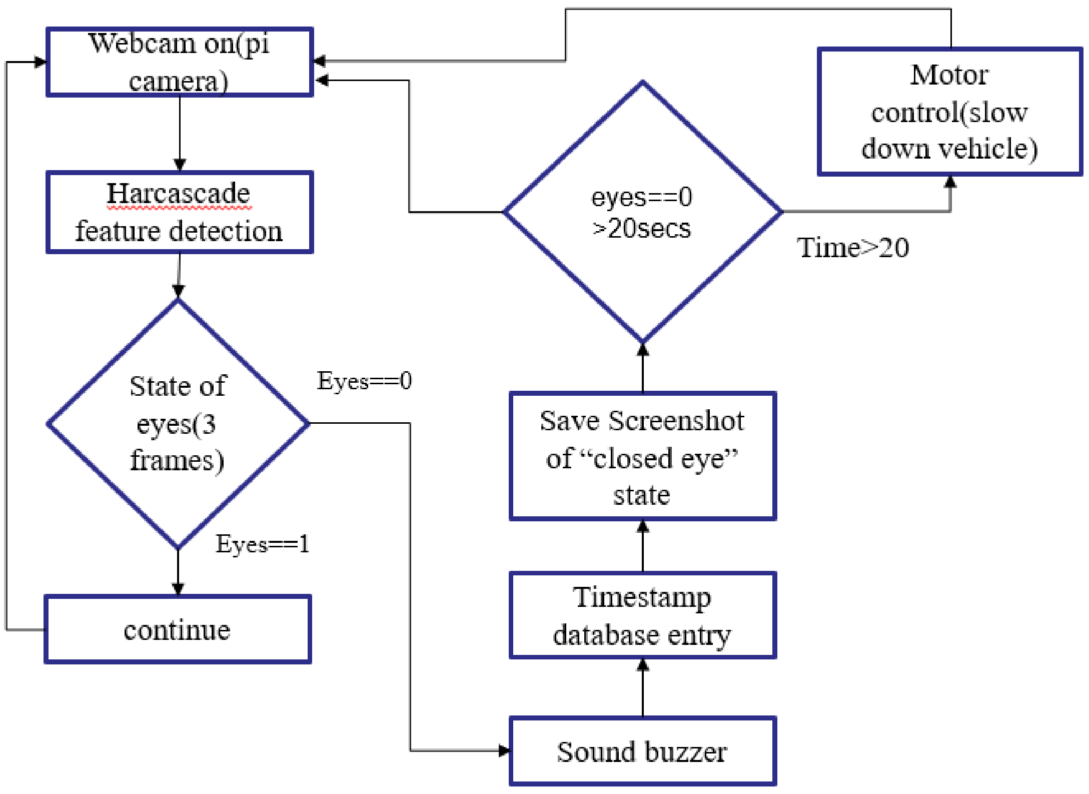


Figure 5.1: Final Block Diagram

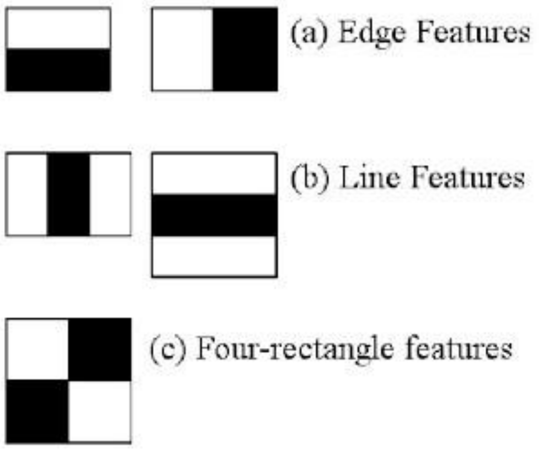


Figure 5.2: Rectangle features [from viola jones rapid object detection paper]

Above image shows the features, that is just like convolution kernel and used to calculate features from the image. Feature is calculated by calculating the difference between the sum of pixel value of two region, black region and white region. However this is a computationally heavy task. Even for an image of size 24x24, it will calculate 160000+ features. To overcome this problem Author had use Adaboost.While training all the features are applied to all the training images, containing positive and negative samples of faces. Error of each feature is calculated and the feature with the minimum error is selected. It brings down the 160000+ features to 6000 good features.Now there is one more problem. In an image, most of the area is non-face and we will spend unnecessary time to apply features to these area. To overcome it, **Cascade of Classifier** is used. So instead of applying all the features at once, a group of features is applied in multiple stage. If one stage pass, then only second stage features are applied, otherwise, area is discarded.

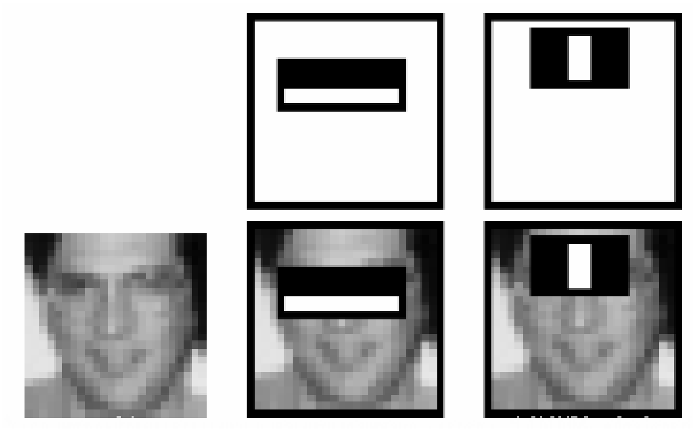


Figure 5.3: The first and second features selected by AdaBoost.
The two features are shown in the top row and then overlayed on a typical training face in the bottom row. [from viola jones rapid object detection paper].Hopefully for face detection, we don't need to train the model from scratch. OpenCV comes with a trainer as well as detector. If you want to train your own classifier for any object like car, planes etc. you can use OpenCV to create one.

Chapter 6 METHODOLOGY

6.1 Object Detection

Object detection is commonly defined as method for discovering and identifying the existence of objects of a certain class. Also it can be considered as a method in image processing to find out an object from images. There are several ways to classify and find objects in a frame. Out of that one way can be based on color identification. But it is not an efficient method to detect the object as several different size object of same color may be present. Hence a more efficient way is Haar-like features, developed by Viola and Jones on the basis of the proposal by Papageorgiou et. al in 1998. Haar-like features are digital image features used in object detection. Or we can say that these are rectangle shaped dark and light areas having similar kind of features like our face. The cascade classifier comprises of a number of stages, where each stage consists of many weak features. The system detects objects by moving a window over the entire image and by forming a strong classifier. The output of each stage is labeled as either positive or negative– positive meaning that an object was found and negative means that the specified object was not found in the image.

6.2 Face detection

We know that face is also a type of object. So we can consider detection of face as a particular case of object detection. In this type of object type of class detection, we try to know where the objects in the interest image are located and what is their size which may belongs to a particular class. The work of algorithm that is made for face detection is mostly concentrated on finding the front side of the face. But the algorithm that are developed recently focus on more general cases. For our case it may be face in the tilted position or any other portion of the faces and also it finds the possibility of multiple faces. Which means the rotation axis with respect to the present observer from the reference of face in a particular. Or even if there is vertical rotation plane then also it is able to solve the purpose. In new type of algorithm it is considered that the picture or video is a variable which means that different condition in them like hue contrast may change its variance. The amount of light may also affect. Also the position of the input may vary the output. Many calculations actualize the face-detection assignment as a two way pattern-differentiation task. It means the contextual features present in the interest image are repeatedly changed into features and this results in preparing the respective classifier on the reference faces which decides if the specified area is a face or any other objects. If we obtain a positive response for the detecting of a face then the process goes for next stage continuation otherwise the algorithm is designed in such manner to go for capturing of image till any hint of face is found. The main algorithm used for this process is Viola Jones algorithm. For getting particular output the utilization of cascade part of open cv is made. Cascade file of OpenCV contains 24 stages and has got 2913 weak classifiers. Its window starts with size of 24 x 24 pixels. Set up for the starting scale has to be made 1.0 and the step size of each scale was set to 1.1 and the position step size Δ was set to 1.0. The total number of scales used is 32 resulting in a total of more than 1.8 million possible detection window which is huge. Training of cascade was done by OpenCV hence it is easy to use.

6.3 Eye Detection

Poor contrast of eyes generally creates a lot of problems in its detection. After successful detection of face eye needs to be detected for further processing. In our method eye is the decision parameter for finding the state of driver. Though detection of eye does not look complex, the actual process is quite hectic. In this case it performs the detection of eye in the specified region with the use of feature detection. Generally, Eigen approach is used for this process. It is a time taking process. When eye detection is done then the result is matched with the reference or threshold value for deciding the state of the driver. Eye detection is divided into two categories: eye contour detection and eye position detection. Basically, eyes are detected based on the assumption that they are darker than other part of the face. Hence Haar Features of similar type can be moved throughout the upper part of the face to match with the feature of eye leading to location of eye. We consider as potential eye areas, the non-skin locales inside face district. Clearly, eyes ought to be inside a face area and eyes are not distinguished as skin by the skin identifier. In this way, we need to discover eye-simple sets among fewer potential eye regions. In recent years several eye detection methods have been developed. Deformable template is one of the popular method in identifying the human eye. In this method, a model of eye is designed first and then eye position is obtained by recursive method. But this method strongly depends on initial position of the eye which should be near the actual position of eye. In the template matching aspect, the proposed algorithm is based on eigenfeatures and neural networks for the extraction of eyes using rectangular fitting from gray-level face images. This method does not need a large set of training images in its advantage and does by eigenfeatures and sliding window. But this algorithm fails if the user uses glasses or having beard. We know that using Haar features in AdaBoost results in increasing computational efficiency and accuracy than other methods for face detection. But Haar feature has a limitation i.e. discriminant capability. Although the Haar features vary with different patterns, sizes and positions, they can only represent the regular rectangular shapes. But for our case of eye detection eye and iris is of round shape. Hence eyes can be represented by learning discriminate features to characterize eye patterns. So an approach towards probabilistic classifier to separate eyes and non-eyes are much better option for better accuracy and for robustness.

Chapter 7 MACHINE LEARNING MODEL

7.1 Transfer Learning Model

Transfer learning generally refers to a process where a model trained on one problem is used in some way on a second related problem.In deep learning, transfer learning is a technique whereby a neural network model is first trained on a problem similar to the problem that is being solved. One or more layers from the trained model are then used in a new model trained on the problem of interest.

This is typically understood in a supervised learning context, where the input is the same but the target may be of a different nature. For example, we may learn about one set of visual categories, such as cats and dogs, in the first setting, then learn about a different set of visual categories, such as ants and wasps, in the second setting.Transfer learning has the benefit of decreasing the training time for a neural network model and can result in lower generalization error.

The weights in re-used layers may be used as the starting point for the training process and adapted in response to the new problem. This usage treats transfer learning as a type of weight initialization scheme. This may be useful when the first related problem has a lot more labeled data than the problem of interest and the similarity in the structure of the problem may be useful in both contexts

7.2 Pre-trained Model Approach

Select Source Model:- A pre-trained source model is chosen from available models. Many research institutions release models on large and challenging datasets that may be included in the pool of candidate models from which to choose from.
Reuse Model:- The model pre-trained model can then be used as the starting point for a model on the second task of interest. This may involve using all or parts of the model, depending on the modeling technique used.

Tune Model:- Optionally, the model may need to be adapted or refined on the input-output pair data available for the task of interest.

7.3 Transfer Learning with Image Data

It is common to perform transfer learning with predictive modeling problems that use image data as input. This may be a prediction task that takes photographs or video data as input. For these types of problems, it is common to use a deep learning model pre-trained for a large and challenging image classification task such as the ImageNet 1000-class photograph classification competition. The research organizations that develop models for this competition and do well often release their final model under a permissive license for reuse. These models can take days or weeks to train on modern hardware. These models can be downloaded and incorporated directly into new models that expect image data as input.

ImageNet:-ImageNet is a dataset consisting of more than 14 million images belonging to nearly 1000 classes.

Three examples of models of this type include:

1)**Oxford VGG Model**

VGG stands for Visual geometry group it is a standard deep convolutional neural network(CNN) architecture with multiple layers. The deep refers to the number of layers with VGG-16 consisting of 16 convolutional layers. The VGG architecture is the basis of ground breaking object recognition models developed as a deep neural network, the VGGNet also surpasses baselines on many tasks and datasets beyond ImageNet.

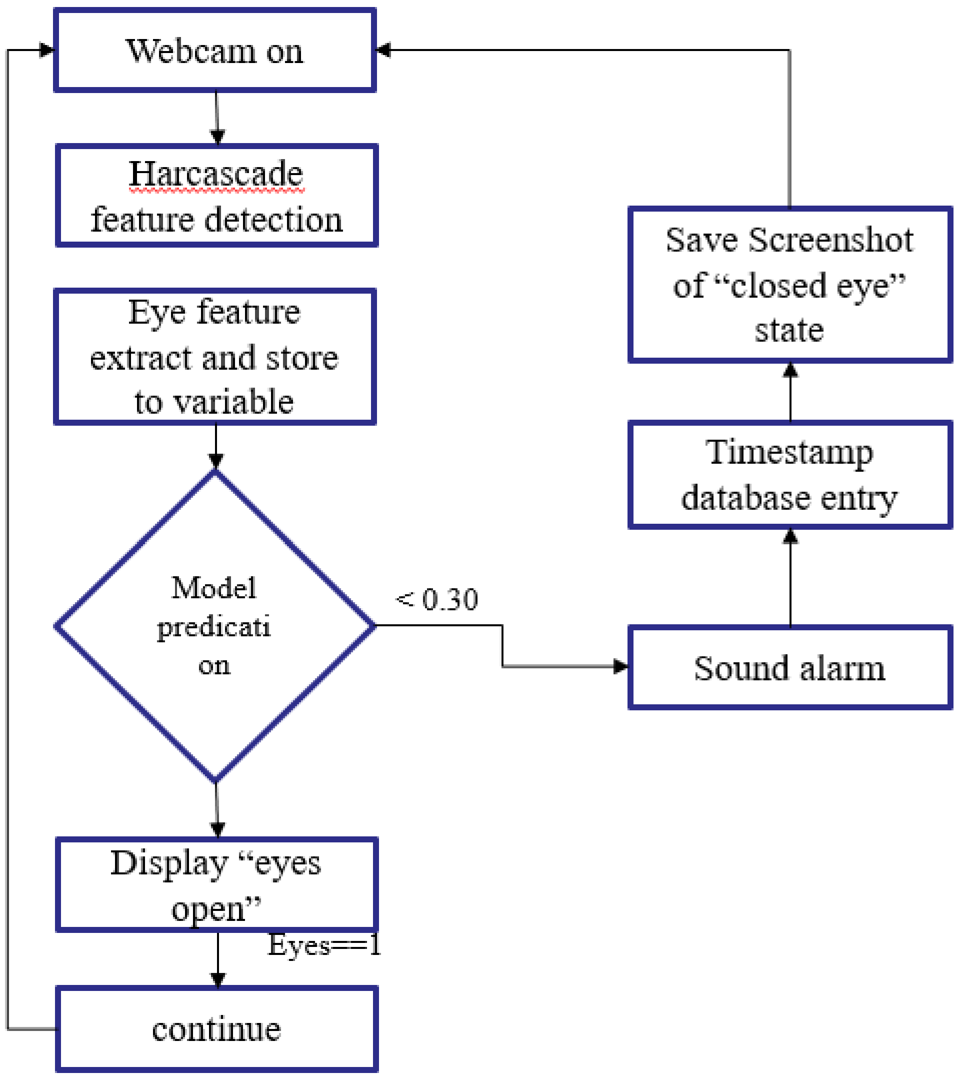
2)**Google Inception Model**

Inception v3 is an image recognition model that has been shown to attain greater than 78.1% accuracy on the ImageNet dataset.

3)**Microsoft ResNet Model**

This approach is effective because the images were trained on a large corpus of photographs and require the model to make predictions on a relatively large number of classes, in turn, requiring that the model efficiently learn to extract features from photographs in order to perform well on the problem.

7.4 ML based prediction model block diagram



Webcam on
Harcascade feature detection
Eye feature extract and store to variable
Model predication
< 0.30
Sound alarm
Timestamp database entry
Save Screenshot of "closed eye" state
Display "eyes open"
Eyes==1
continue

Figure 7.1: ML based prediction model block diagram

Chapter 8 RESULTS AND DISCUSSIONS

8.1 Raspberry pi simulation result:

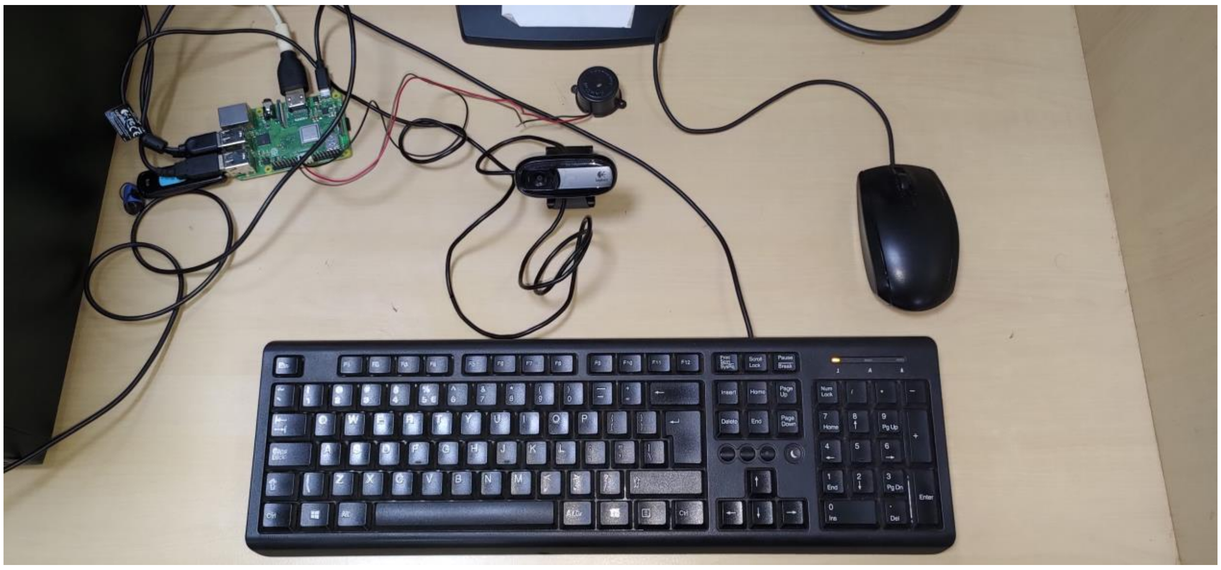
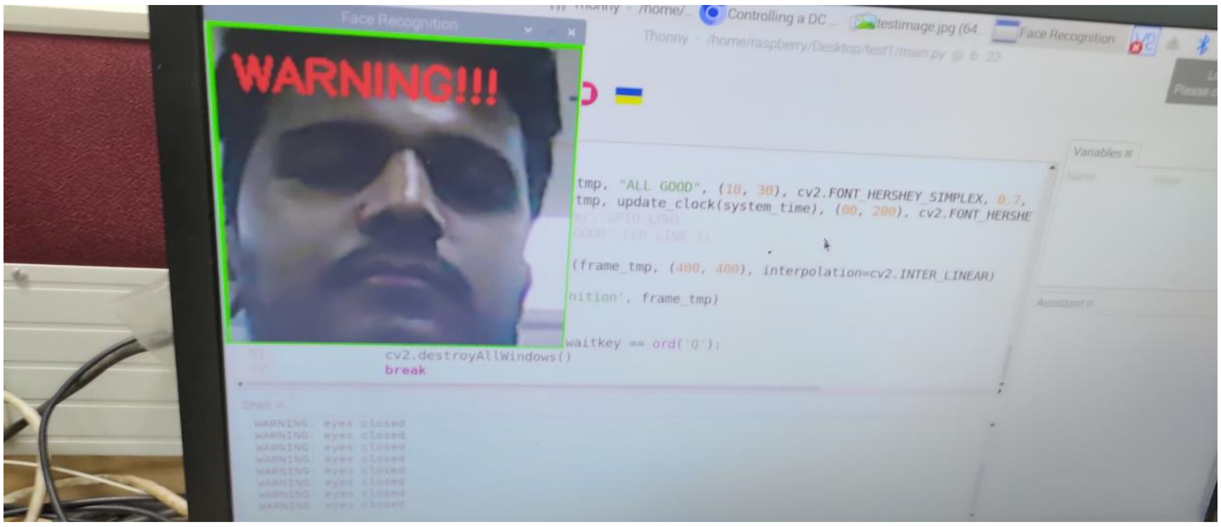


Figure 8.1: Raspberry Pi hardware simulation



```
tmp, "ALL GOOD", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7,
tmp, update_clock(system_time), (00, 200), cv2.FONT_HERSHE

(frame_tmp, (400, 400), interpolation=cv2.INTER_LINEAR)
nition', frame_tmp)

waitkey == ord('Q'):
        cv2.destroyAllWindows()
        break
```

```
WARNING: eyes closed
WARNING: eyes closed
WARNING: eyes closed
WARNING: eyes closed
WARNING: eyes closed
WARNING: eyes closed
WARNING: eyes closed
WARNING: eyes closed
```

Figure 8.2: Raspberry simulation result on Monitor

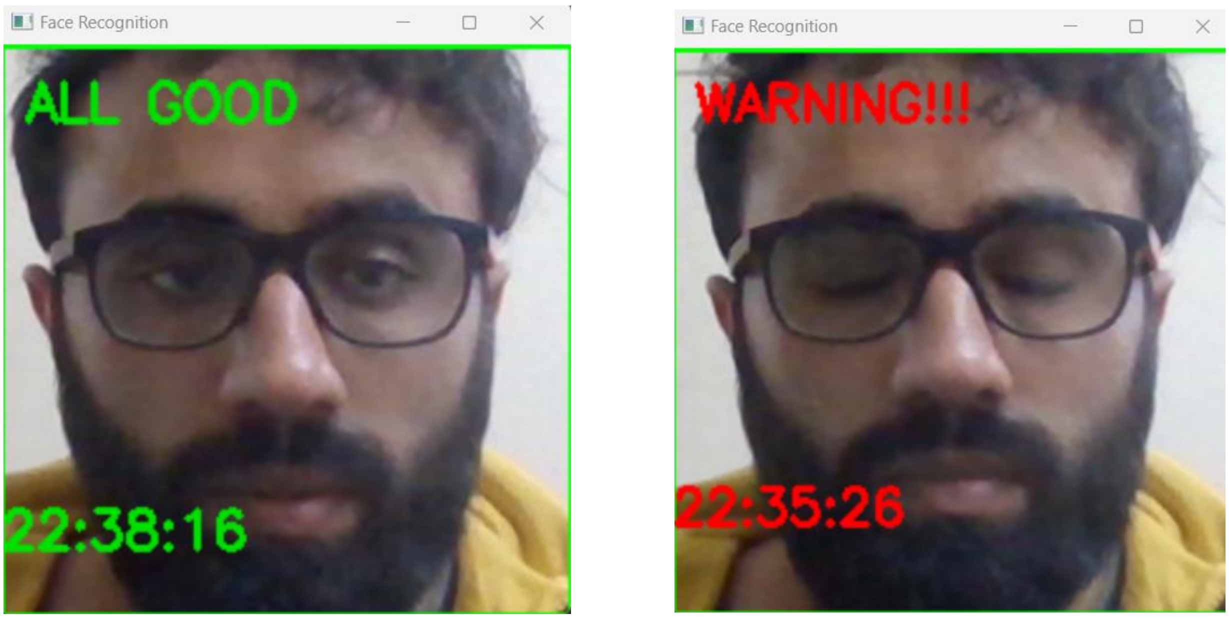
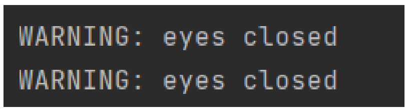
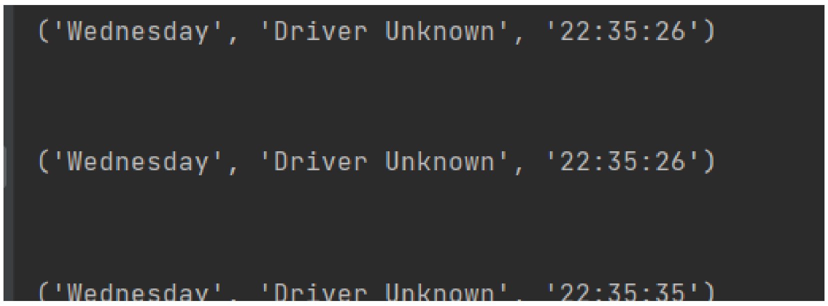


Figure 8.3: Simulation result with timestamp



```
WARNING: eyes closed
WARNING: eyes closed
```

Figure 8.4: Console Warning message



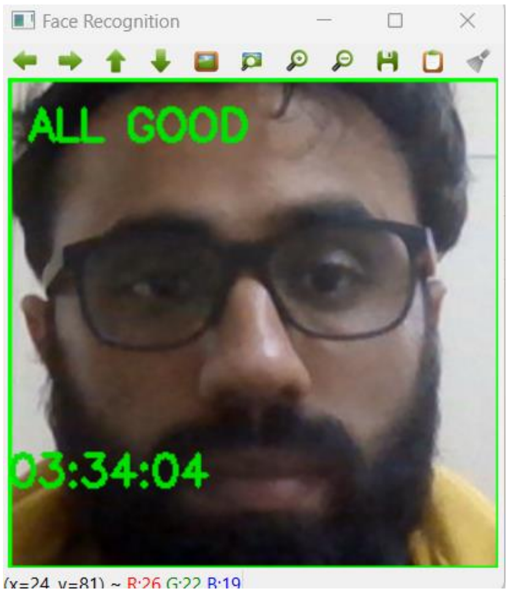
```
('Wednesday', 'Driver Unknown', '22:35:26')


('Wednesday', 'Driver Unknown', '22:35:26')


('Wednesday', 'Driver Unknown', '22:35:35')
```

Figure 8.5: Database message

8.2 ML prediction model implementation result



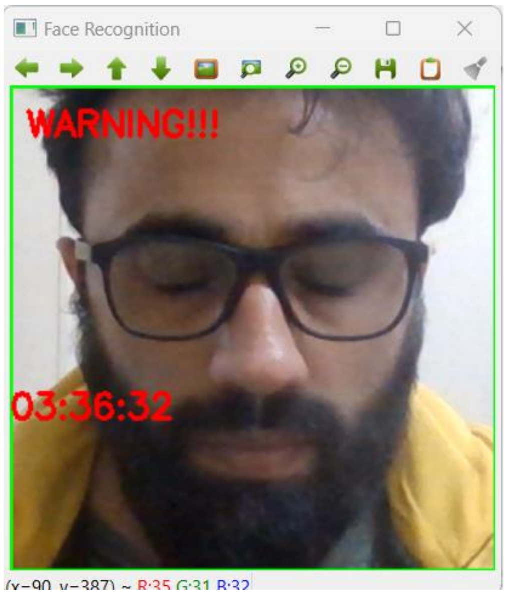
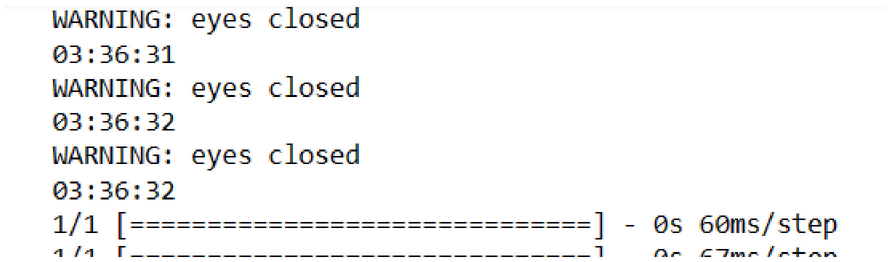


Figure 8.6: ML prediction model simulation result



```
WARNING: eyes closed
03:36:31
WARNING: eyes closed
03:36:32
WARNING: eyes closed
03:36:32
1/1 [==============================] - 0s 60ms/step
1/1 [==============================] - 0s 67ms/step
```

Figure 8.5 Warning message on ML model output

```
('Thursday', 'Driver Unknown', '03:36:31')

('Thursday', 'Driver Unknown', '03:36:31')

('Thursday', 'Driver Unknown', '03:36:32')

None
```

Figure 8.7 Database message

- The driver's eyes are located, monitored, and used to quickly detect tiredness as part of the real-time drowsiness detection system.
- This method enables the machine to recognize closed eyes (sleep condition).
- After determining the sleep condition, the real-time system sounds the alarm as a warning.
- A database is used to stored the timestamps of closed eye state and takes the screenshot and save the sleep state as a jpg file(may be used as for new dataset/ImageNet).
- The implementation of an machine learning model, increases the accuracy of detection.

Chapter 9 BILL OF MATERIALS

Components	**Rupees (₹)**
Raspberry Pi 3 Model-B+ with 32 GB RAM	9000
Piezo buzzer	200
Web camera	800
Total	10000

Table 10.1: Bill of Materials

Chapter 10 CODES IMPLEMENTED

10.1 Main.py

```
import datetime
import cv2
from pygame import mixer
from sqllite import insert_timestamp
from timestamp import Time
from peripherals import main


mixer.init()
sound = mixer.Sound('C:/Users/visha/OneDrive/Desktop/ES/mini project/mixkit-vintage-
warning-alarm-990.wav')
faceCascade = cv2.CascadeClassifier('C:/Users/visha/OneDrive/Desktop/ES/mini
project/haarcascade_frontalface_alt.xml')
eyeCascade = cv2.CascadeClassifier('C:/Users/visha/OneDrive/Desktop/ES/mini
project/haarcascade_eye_tree_eyeglasses.xml')
count=0


def update_clock(system_t):
    os = datetime.datetime.now()
    system_t = os.strftime("%H:%M:%S")
    # system_d = os.strftime('%A')
    return system_t

def update_day(system_d):
    day= datetime.datetime.now()
    system_d = day.strftime('%A')
    return system_d

ostime = datetime.datetime.now()

system_time=ostime.strftime('%H:%M:%S')
system_day=ostime.strftime('%A')

cap = cv2.VideoCapture(0)
while(1):

    ret, img = cap.read()
    if ret:
        frame = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = faceCascade.detectMultiScale(frame, 1.1, 5)
        if len(faces) > 0:
            for (x, y, w, h) in faces:
                cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
            frame_tmp = img[faces[0][1]:faces[0][1] + faces[0][3], faces[0][0]:faces[0][0] +
faces[0][2]:1, :]
            frame = frame[faces[0][1]:faces[0][1] + faces[0][3], faces[0][0]:faces[0][0] +
faces[0][2]:1]
            eyes = eyeCascade.detectMultiScale(frame, scaleFactor=1.1, minNeighbors=5,
minSize=(30, 30))
            if len(eyes)==0:
                count+=1
                if(count==3):
                    # Saved closed eye image in folder
                    cv2.imwrite('C:/Users/visha/OneDrive/Desktop/ES/mini
project/testimage.jpg', img)
                    count = 0
                    # lcd_string("WARNING",LCD_LINE_1)
                    # lcd_string("Drowsy",LCD_LINE_1)
                    # lcd_string("detected",LCD_LINE_2)
                    print('WARNING: eyes closed')
                    a = Time(update_day(system_day), 'Driver Unknown',
update_clock(system_time))
                    # created database for saving timestamps
                    insert_timestamp(a)
                    cv2.putText(frame_tmp, "WARNING!!!", (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.7,(0, 0, 255), 2)
                    cv2.putText(frame_tmp,
update_clock(system_time),(00,200),cv2.FONT_HERSHEY_SIMPLEX,0.7,(0,0,255),2)
                    # GPIO.output(buzzer, GPIO.HIGH)
                    sound.play(0, 5, 0)
            else:
                count = 0
                cv2.putText(frame_tmp, "ALL GOOD", (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 225, 0), 2)
                cv2.putText(frame_tmp, update_clock(system_time), (00, 200),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 225, 0), 2)
                # GPIO.output(buzzer, GPIO.LOW)
                # lcd_string("ALL GOOD",LCD_LINE_1)

            frame_tmp = cv2.resize(frame_tmp, (400, 400),
interpolation=cv2.INTER_LINEAR)

            cv2.imshow('Face Recognition', frame_tmp)

        waitkey = cv2.waitKey(1)
        if waitkey == ord('q') or waitkey == ord('Q'):
            cv2.destroyAllWindows()
            break
```

Chapter 11 SOFTWARE AND HARDWARE DETAILS

11.1 Software Details

11.1.1 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

11.1.2 OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. The library is used extensively in companies, research groups and by governmental bodies.OpenCV's deployed uses span the range from stitching street view images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with Standard Template Library containers.

11.2. Hardware Description

11.2.1 Raspberry Pi 3 Model B+

The Raspberry pi is a single computer board with credit card size that can be used for many tasks that your computer does, like games, word processing, spreadsheets and also to play HD video. It was established by the Raspberry pi foundation from the UK. It has been ready for public consumption since 2012 with the idea of making a low cost educational microcomputer for students and children. Raspberry pi board is a portable device and The Raspberry Pi 3 Model B+ (Pi3B+) is the first of a new generation of Raspberry Pi computers supporting more RAM and with significantly enhanced CPU, GPU and I/O performance all within a similar form factor, power envelope and cost as the previous generation Raspberry Pi 3B+.

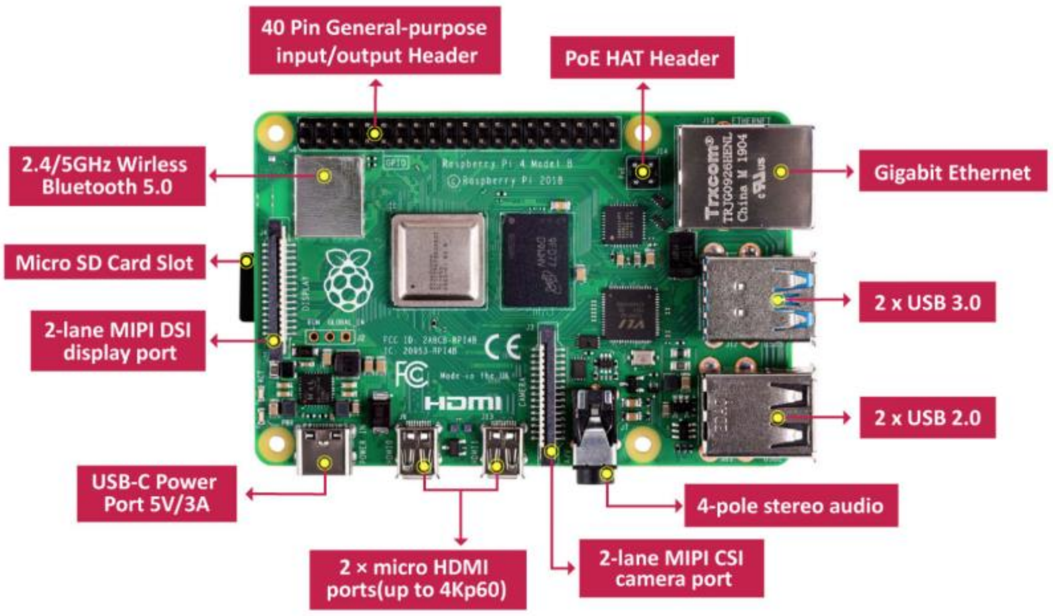

40 Pin General-purpose input/output Header
PoE HAT Header
2.4/5GHz Wirless Bluetooth 5.0
Gigabit Ethernet
Micro SD Card Slot
2 x USB 3.0
2-lane MIPI DSI display port
2 x USB 2.0
USB-C Power Port 5V/3A
4-pole stereo audio
2 × micro HDMI ports(up to 4Kp60)
2-lane MIPI CSI camera port

Figure 11.1: Raspberry Pi 3B+

Features

Hardware

- Quad core 64-bit ARM-Cortex A72 running at 1.5GHz.
- 1, 2 and 4 Gigabyte LPDDR4 RAM options.
- H.265 (HEVC) hardware decode (up to 4Kp60).

- H.264 hardware decode (up to 1080p60).
- Video Core VI 3D Graphics.
- Supports dual HDMI display output up to 4Kp60.

Interfaces

- 802.11 b/g/n/ac Wireless LAN.
- Bluetooth 5.0 with BLE.
- 1x SD Card.
- 2x micro-HDMI ports supporting dual displays up to 4Kp60 resolution.
- 2x USB2 ports.
- 2x USB3 ports.
- 1x Gigabit Ethernet port (supports PoE with add-on PoE HAT).
- 1x Raspberry Pi camera port (2-lane MIPI CSI).
- 1x Raspberry Pi display port (2-lane MIPI DSI).

GPIO supporting various interface options

- Up to 6x UART.
- Up to 6x I2C.
- Up to 5x SPI.
- 1x SDIO interface.
- 1x DPI (Parallel RGB Display).
- 1x PCM.
- Up to 2x PWM channels.
- Up to 3x GPCLK outputs.

Software

- ARMv8 Instruction Set.
- Mature Linux software stack.
- Recent Linux kernel support.
- Stable and well supported userland.
- Availability of GPU functions using standard APIs

Power Requirements

The Pi 3B+ requires a good quality USB-C power supply capable of delivering 5V at 3A. If attached downstream USB devices consume less than 500mA, a 5V, 2.5A supply may be used.

GPIO Interface

The Pi 3B+ makes 28 BCM2711 GPIOs available via a standard Raspberry Pi 40-pin header. This header is backwards compatible with all previous Raspberry Pi boards with a 40-way header.

GPIO Pin Assignments

As well as being able to be used as straightforward software controlled input and output (with programmable pulls), GPIO pins can be switched (multiplexed) into various other modes backed by dedicated peripheral blocks such as I2C, UART and SPI. In addition to the standard peripheral options found on legacy Pi’s, extra I2C, UART and SPI peripheral shave been added to the BCM2711 chip and are available as further mux options on the Pi 3B+. This gives users much more flexibility when attaching add-on hardware as compared to older models.

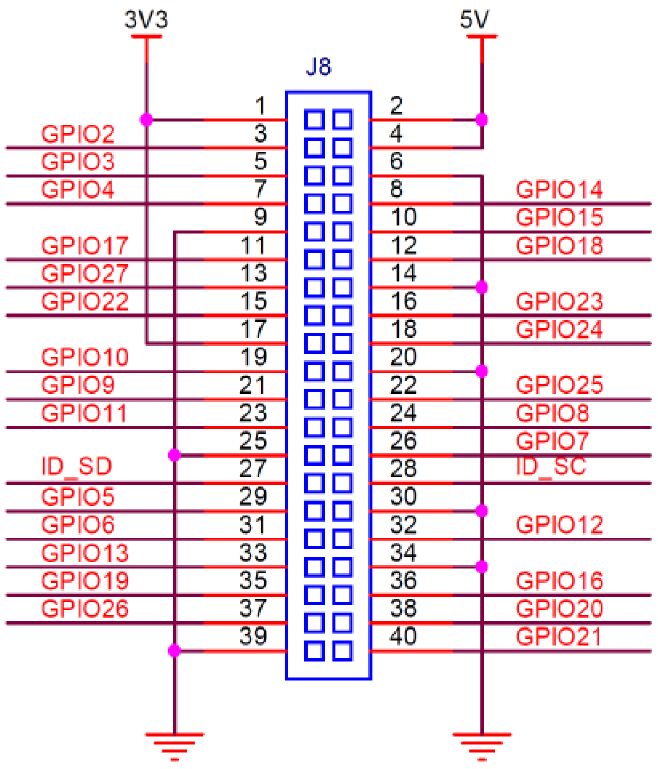


Figure 11.2: Raspberry GPIO pin diagram

Power Requirements

The Pi3B+ requires a good quality USB-C power supply capable of delivering 5V at 3A. If attached downstream USB devices consume less than 500mA, a 5V, 2.5A supply may be used.

Camera and Display Interfaces

The Pi3B+ has 1x Raspberry Pi 2-lane MIPI CSI Camera and 1x Raspberry Pi 2-lane MIPI DSI Display connector. These connectors are backwards compatible with legacy Raspberry Pi boards, and support all of the available Raspberry Pi camera and display peripherals.

USB

The Pi3B+ has 2x USB2 and 2x USB3 type-A sockets. Downstream USB current is limited to approximately 1.1A in aggregate over the four sockets.

HDMI

The Pi3B+ has 2x micro-HDMI ports, both of which support CEC and HDMI 2.0 with resolutions up to 4Kp60.

11.2.2 Mini Piezo Buzzer

Figure 11.3: Piezo Buzzer

DC Operation

Apply 3V to 5V to this piezo buzzer module and you'll be rewarded with a loud 2 KHz BEEP. Unlike a plain piezo, this buzzer does not need an AC signal. Inside is a piezo element plus the driver circuitry that makes it oscillate at 2 KHz. The piezo buzzer is 5V TTL logic compatible and Breadboard friendly pin spacing. This buzzer is ideal when you need to fit a buzzer in a small place. It has its own built-in drive circuit. It offers low current consumption. Used in manufacturing applications such as laptops, alarms, pagers, etc. Great for use as part of a Code Practice Oscillator. The piezo buzzer is polarized, meaning that power must be applied to the correct pins. On the top of the case, there is a plus (+) sign to indicate the anode connection pin.

Specifications

- Operation Voltage: 3-5V DC.
- Current: <25mA.
- SPL: 85dBA/10cm.
- Frequency: 2,300Hz.
- Colour: Black.
- Operating Temperature: - 20° to +65°C.
- Weight: 2.4 gram.
- Size: 1.2cm diameter x 1cm tall (0.47" x 0.39").

- Pin Spacing: 7.6mm.

11.2.3 Web camera

Figure 11.4: Webcam

A webcam is a small digital video camera directly connected to Raspberry pi board. Webcams can be accessed through python software in raspberry pi.Webcams are capable of taking pictures as well as high-definition videos, although the video quality can be lower compared to other camera models.

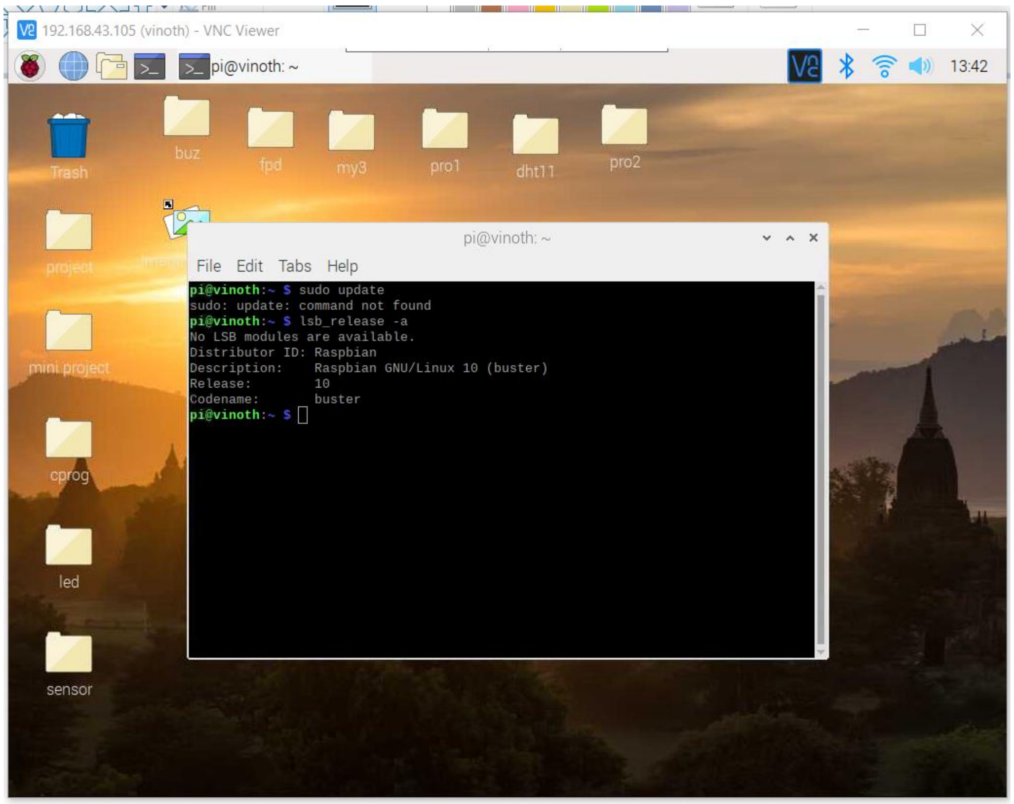
Specifications

- Bright Led.
- Manual focus.
- Powered by USB and no external power supply needed.
- USB 2.0.
- 2Mpix (1600*1200).
- Night vision capabilities.

Chapter 12 SOFTWARE INSTALLATION STEPS

12.1 Raspbian operating system

Latest version (Raspbian Version 10) of raspbian os is downloaded from official raspberry pi website (https://www.raspberrypi.org/) and installed on raspberry pi 4 model B+ board.

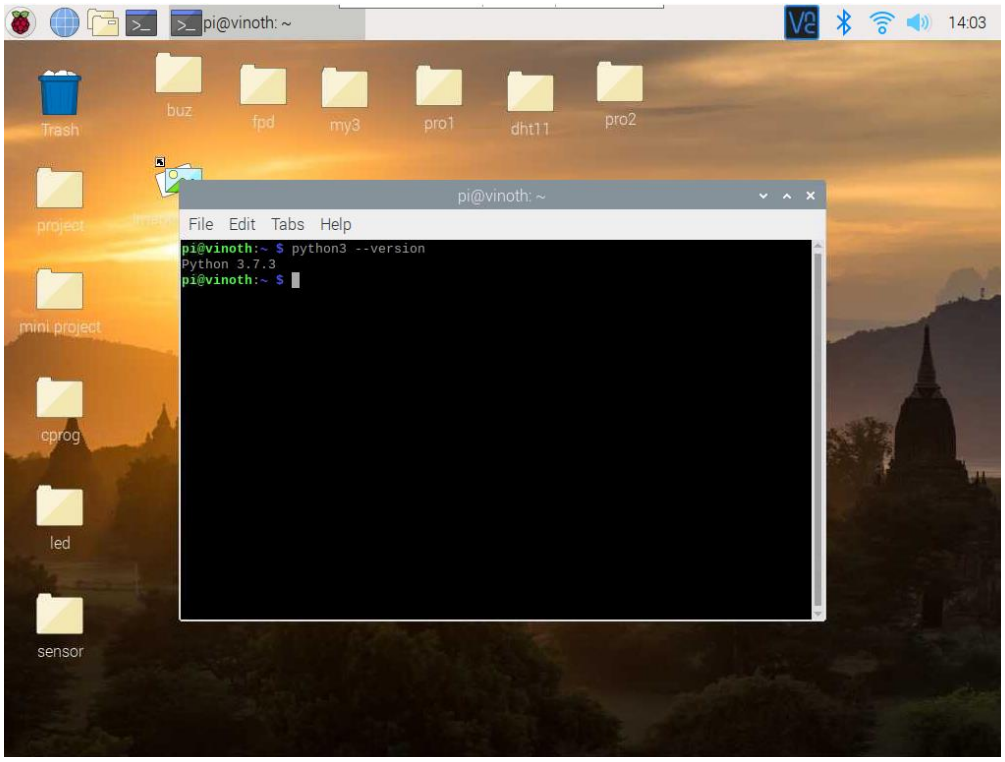


```
pi@vinoth:~ $ sudo update
sudo: update: command not found
pi@vinoth:~ $ lsb_release -a
No LSB modules are available.
Distributor ID: Raspbian
Description:    Raspbian GNU/Linux 10 (buster)
Release:        10
Codename:       buster
pi@vinoth:~ $
```

Fig 12.1: Screenshot of Raspbian desktop about raspbian version.

12.2 Python Installation on Raspbian operating system Install command

```
sudo apt-get install python3
```



```
pi@vinoth:~ $ python3 --version
Python 3.7.3
pi@vinoth:~ $ 
```

Fig 12.2: Version of python in screenshot

12.3 Installation of OpenCV in Raspbian operating system

1) Update os system:
sudo apt-get update
sudo apt-get upgrade
2) Install dependencies:
sudo apt-get install build-essential cmake pkg-config
sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
sudo apt-get install libxvidcore-dev libx264-dev
sudo apt-get install libgtk2.0-dev libgtk-3-dev
sudo apt-get install libatlas-base-dev gfortran
3) Install Python 3 and Pip3:
sudo apt-get install python3-dev
sudo apt-get install python3-pip
4) Install Opencv:
pip3 install opencv-python
5) Extra depencies for Opencv and the Camera:
sudo apt-get install libqtgui4
sudo modprobe bcm2835-v4l2
sudo apt-get install libqt4-test

Chapter 13 HARDWARE CIRCUIT/INTERFACE DIAGRAM

13.1 Hardware Interface Block Diagram

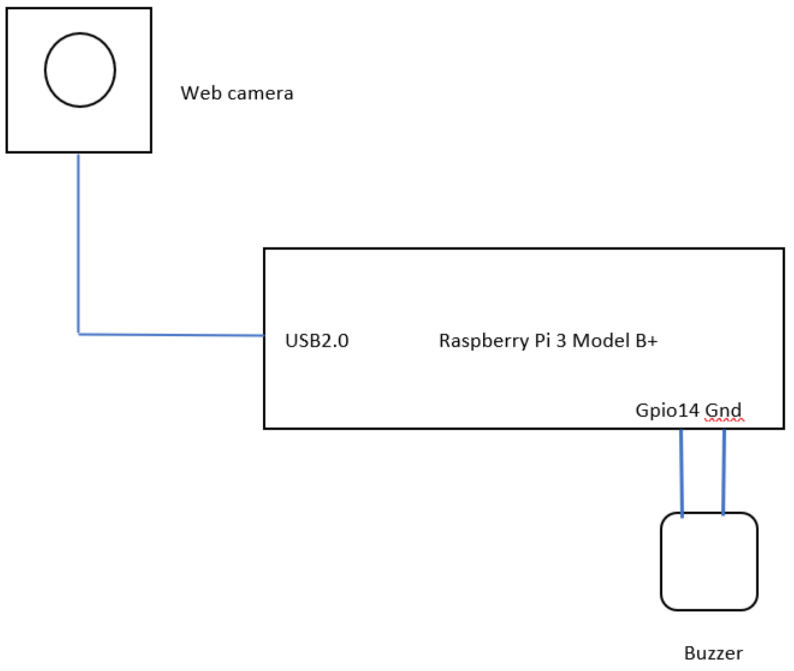


Fig 13.1: Hardware Block Diagram

13.2 Complete hardware setup

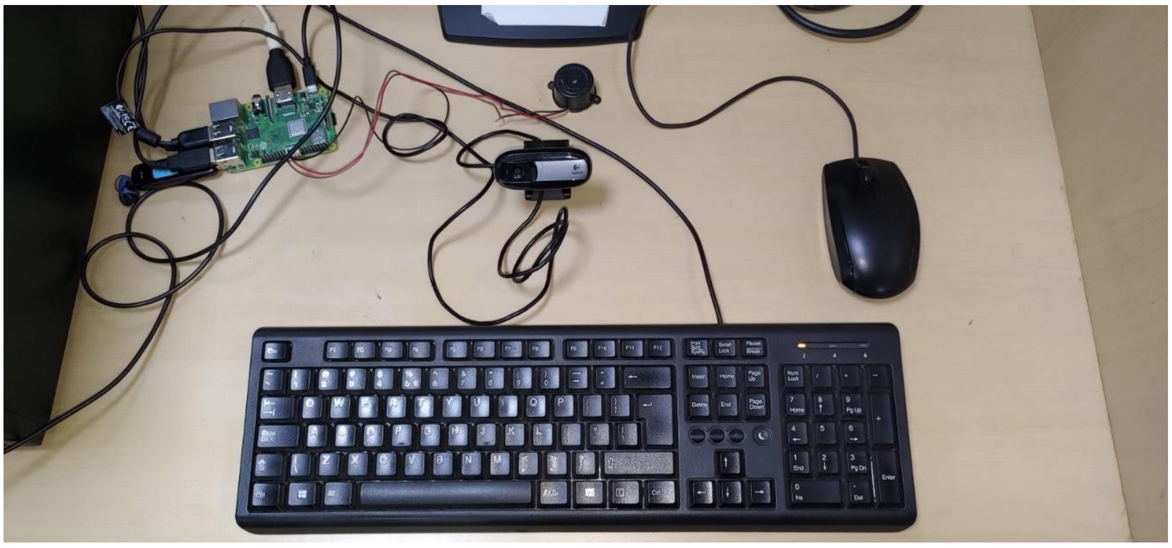


Fig 13.2: Screenshot of Raspbian desktop about raspbian version.

Chapter 14 CONCLUSION AND FUTURE SCOPE

14.1 Conclusions

Implementation of drowsiness detection with Raspberry pi was done with static image, which will check multiple images if the images continuously closed for certain counts then system will ring the alarm. The same method can be implemented on web camera for live videos. Comparing to laptops and personal computers raspberry pi shows slower results.

14.2 Scope for future work

This project is designed for detection of drowsy state of eye and give and alert signal or warning may be in the form of audio or any other means. But the response of driver after being warned may not be sufficient enough to stop causing the accident meaning that if the driver is slow in responding towards the warning signal then accident may occur. Hence to avoid this we can design and fit a motor driven system and synchronize it with the warning signal so that the vehicle will slow down after getting the warning signal automatically. Also we can avoid the use of Raspberry Pi which is not so fast enough for video processing by choosing our own mobile phone as the hardware. This can be done by developing a proper mobile application which will perform the same work as Raspberry pi and response will be faster and effective.

BIBLIOGRAPHY

[1] Danisman, T., Bilasco, I.M., Djeraba, C. and Ihaddadene, N., 2010, October. Drowsy driver detection system using eye blink patterns. In 2010 International Conference on Machine and Web Intelligence (pp. 230-233). IEEE.

[2] Sarkar, D. and Chowdhury, A., 2014. A real time embedded system application for driver drowsiness and alcoholic intoxication detection. International Journal of Engineering Trends and Technology, 10(9), pp.461-465.

[3] Musale, T. and Pansambal, B., 2016. Real Time Driver Drowsiness Detection system using Image processing. Research in Engineering Application & Management (IJREAM), 2(08).

[4] Transfer Learning in Keras with Computer Vision Models

https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/

[5] Musale, T. and Pansambal, B., 2016. Real Time Driver Drowsiness Detection system using Image processing. Research in Engineering Application & Management (IJREAM), 2(08).

[6] Seemar, P. and Chandna, A., INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY DROWSY DRIVER DETECTION USING IMAGE PROCESSING.

[7] Danisman, T., Bilasco, I.M., Djeraba, C. and Ihaddadene, N., 2010, October. Drowsy driver detection system using eye blink patterns. In 2010 International Conference on Machine and Web Intelligence (pp. 230-233). IEEE.

[8] Batchu, S. and Kumar, S.P., 2015. Driver Drowsiness Detection to Reduce the Major Road Accidents in Automotive Vehicles. International Research Journal of Engineering and Technology, 2, pp.345-349.

[9] Ghosh, A., Chatterjee, T., Samanta, S., Aich, J. and Roy, S., 2017. Distracted driving: A novel approach towards accident prevention. Adv. Comput. Sci. Technol, 10, pp.2693-2705.

PROEJCT TEAM MEMBER DETAILS

NAME	**VISHAK V KOTEMANE**
BRANCH	**Embedded Systems**
REGISTRATION NUMBER	**221039002**
MOBILE NUMBER	**7022284574**
EMAIL ID	**vishak.msismpl2022@learner.manipal.edu**

NAME	**SANDESH B S**
BRANCH	**Embedded Systems**
REGISTRATION NUMBER	**221039004**
MOBILE NUMBER	**9449374347**
EMAIL ID	**sandesh.msismpl@learner.manipal.edu**

NAME	**Mohan Kumar J**
MOBILE NUMBER	**9901728982**
EMAIL ID	**mohujs@gmail.com** **mohan.js@manipal.edu**

*