VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE ENGINEERING

DRIVER DROWSINESS DETECTION SYSTEM

Team 13

HARSH VIVEK LONDHEKAR (19BCE0496)
RITIK SINGH (19BCE2255)
MOHAMED ARSHAD PEERMOHAMED (19BCE0596)

PROJECT REPORT
OF
CSE - 3009 INTERNET OF THINGS

FALL SEMESTER 2021-22

SUBMITTED TO
FACULTY: Dr. VISHNU SRINIVASA MURTHY Y
SLOT : B1

CERTIFICATE

This is to certify that Harsh Vivek Londhekar (19BCE0496), Mohamed Arshad Peermohamed (19BCE0596), Ritik Singh (19BCE2255) 3rd year B.Tech, (Computer Science & Engineering) from Vellore Institute of Technology (VIT) has successfully completed their project work in the field of Internet of Things (IoT) on the topic Driver Drowsiness Detection System. This is a record of his/her own work carried out during the Fall Semester of the Academic Year 2021-22 under the guidance of Dr. Vishnu Srinivasa Murthy Yarlagadda. He has presented his project in the presence of faculty.

Dr. Vishnu Srinivasa Murthy Y,

Assistant Professor / Guide

ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of our project. All that we have done is only due to such supervision and assistance and we would not forget to thank them.

First and foremost, we owe our deep gratitude to our IoT professor Dr. Vishnu Srinivasa Murthy Y, who took keen interest in our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a good system. We will also not forget to mention our group participants and also friends for the viable information which they provided to us during the course of this project which ultimately lead to amelioration of this project.

Last but not the least, we would also like to thank VIT University, and lectures with the help of which we were able to grasp knowledge for making this project. Also, we would like to thank our classmates and friends who helped us complete this project in such a short time.

Harsh Vivek Londhekar

Mohamed Arshad Peermohamed

Ritik Singh

December 2021

I. Abstract

Accidents are very common these days due to casual driving by civilians. An endless number of people including truck, taxi and bus drivers lose their lives due to drowsiness. Hence, there is a need for a computing device to display the drowsiness stage of drivers and alert drivers as soon as it predicts the possibility of such occurrences. So in this project, we advocate a gadget regarded as a driver drowsiness detection device primarily based on behavioral measures and the usage of machine learning techniques. There are many facial elements that can be extracted from the face and eyes to infer the stage of drowsiness. These encompass head movements, eye blinks and yawning. The undertaking contains the driver monitoring device resulting in a whole protected force ensuring module. Driver Monitoring System would continuously seize the photograph of the eyes of the drivers using Opencv and Machine learning models. In the driver monitoring system, drowsiness score and yawning rate of drivers will be calculated based on the extracted facial features. Since Pulse rate is a critical issue in determination of drowsiness, the pulse rate of the drivers will be calculated using a pulse sensor tied to the driver's wrist. This pulse price will be sent to the cloud using Arduino IDE through Node MCU. The mission consists of Central Cloud servers which monitor the heart rate and plot its corresponding sketch using the assistance of ThingSpeak. The Central Monitoring System will continuously screen the drowsiness score, yawning score and pulse price of the drivers. If any of these parameter values is less than threshold value, then an update will be sent to the Driver and an alarm will ring.

II. Introduction

There is full-size statistical proof that points to driver drowsiness as a main cause of road accidents all over the world. Driving for prolonged intervals of time can lead to accidents if rest is no longer taken.Drowsiness is an important purpose of vehicular accidents. The World Health Organization (WHO) has shown that South Africa among African regions has the highest road visitors accident fatalities of about 26.6 p.c per 1,00,000 populace. According to surveys, the price of accidents is extra throughout midnight or early daytime. This occurs because drivers frequently tend to fall asleep or lose their concentration in such hours. This is the direct effect of the sleep cycle of human beings and cannot be constantly prevented. Drowsiness can be described as an organic country where the physique is in-transition from an awake state to a snoozing state. At this stage, a driver can lose concentration and be unable to perform actions like heading off head-on collisions or braking timeously. The following signs and symptoms indicate that the driver is drowsy.

1. Frequently yawning.
2. Inability to keep eyes open.
3. Lowering of Pulse rate.

There are various measures which indicate the level of drowsiness. These measures can be grouped into three categories: I. Physiological Measures, II. Vehicle-based Measures, and III. Behavioral Measures

In Physiological measures, digital devices are inserted onto the skin of the drivers through which bought measurements access the driver’s conditions.

These devices are Electroencephalography (EEG), Electrocardiography(ECG) and Electrooculogram (EOG) . These types of gadgets yield surprisingly correct effects but they have practical limitations. In the case of the 2d category, vehicle management structures are used which helps to determine the driver's drowsiness. This car manipulator machine consists of guidance wheel movements, braking patterns, and measurements of lane departure. Steering wheel measurements is one of the methods which yield higher results than other vehicle-based methods. Vehicle-based methods are now not as reliable in detecting drowsiness accurately due to the fact they are dependent on the nature of the road and the driver's driving skills. The third class is behavioral measures which are also acknowledged as laptop imaginative and prescient measures. This method depends on the man or woman rather than the automobile so they have a tendency to be extra reliable as in contrast to others. This is more practical than different drowsiness measurements methods. Since the face carries a lot of data, cameras detect the facial aspects that can be extracted from the face to infer the degree of drowsiness. As they are non-invasive in nature, they are becoming a famous way of detecting drowsiness as they are non-invasive in nature. This venture demonstrates three one-of-a-kind non-invasive methods of drowsiness detection particularly Eye-drowsiness Detection the usage of Opencv and Aspect Ratio, Yawning detection using Euclidean distance between each lips, Pulse Rate Measurement.

III. Motivation

Accidents are very common these days due to casual driving by civilians. A countless number of people including truck, taxi and bus drivers lose their lives due to drowsiness. Hence, there is a need for a machine to monitor the drowsiness level of drivers and alert drivers as soon as it predicts the possibility of such occurrences. Driver drowsiness detection system helps to avoid accidents caused by drowsiness by advising drivers to take a break in time.

So in this project, we propose a system known as driver drowsiness detection system based on behavioral measures using machine learning techniques.

IV. Literature Survey

[1]

Approach : DriCare System uses multiple Convolutional NeuralNetworks(CNN)-KCF (MC-KCF), which optimizes KCF algorithm and CNN to assess the state of the eye

Remarks : The experimental results show that when the driver is awake, the blinking frequency and eye-closing time are low. The MC-KCF algorithm demonstrates the best tracking accuracy. If the driver wears glasses and the driving environment is slightly dim, the accuracy of fatigue driving is reduced.

Accuracy of the results shown in the given paper were calculated to be 92% accurate.

[2]

Approach : The following algorithms were used in the given paper, Perclos, Camshift algorithm, Haar Training, Viola Jones Algorithm.

Remarks: Input is captured by a camera, processed by the Raspberry-pi module, and the output is in the form of a buzzer that alerts the user, as and when drowsiness is detected.

All the 4 approaches are found to pretty accurate

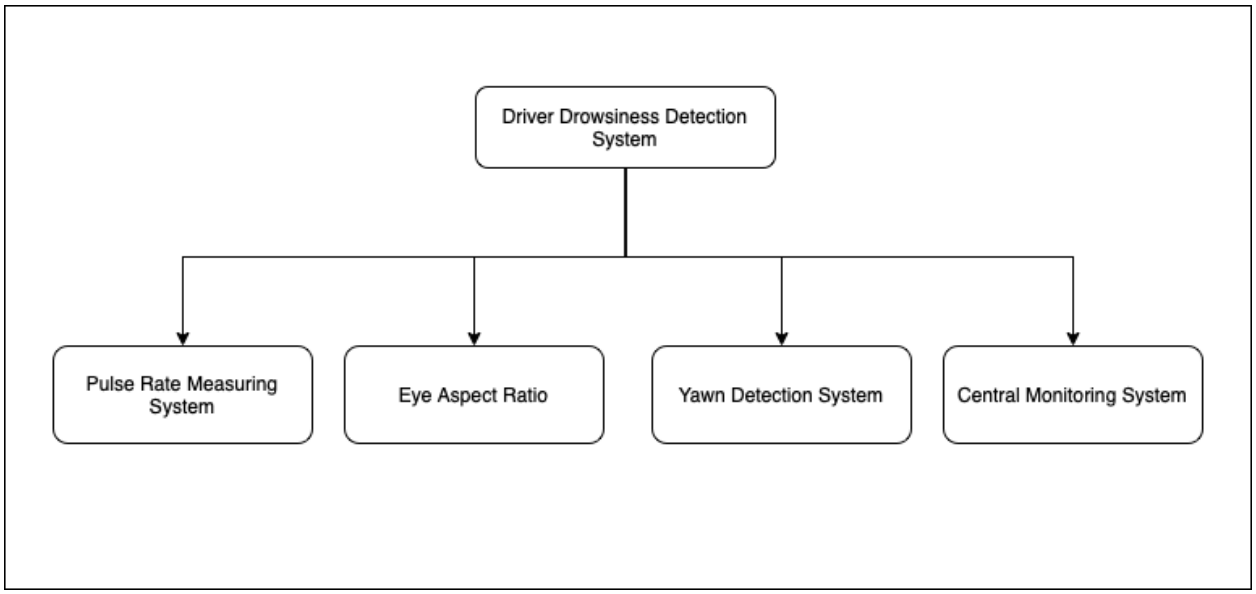
[3]

Approach : Methods used in the paper were Support Vector Machine (SVM), Hidden Markov Model (HMM), and Convolutional Neural Network (CNN)

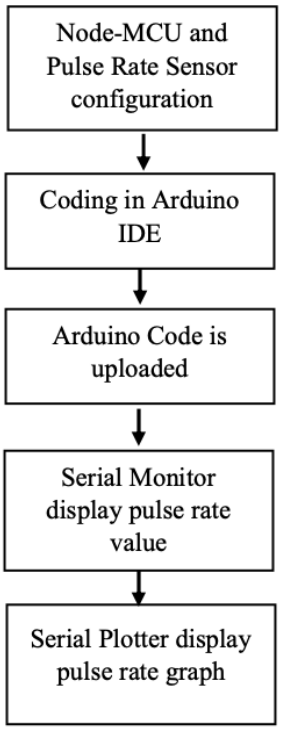
Remarks : Analysis highlighted the performance of CNNs, which outperformed other approaches, but also showed that there is a need for larger datasets and standard benchmarking measures for drowsiness detection.

In the paper it was found that among all the papers used CNN has the most accuracy.

V. Methodology

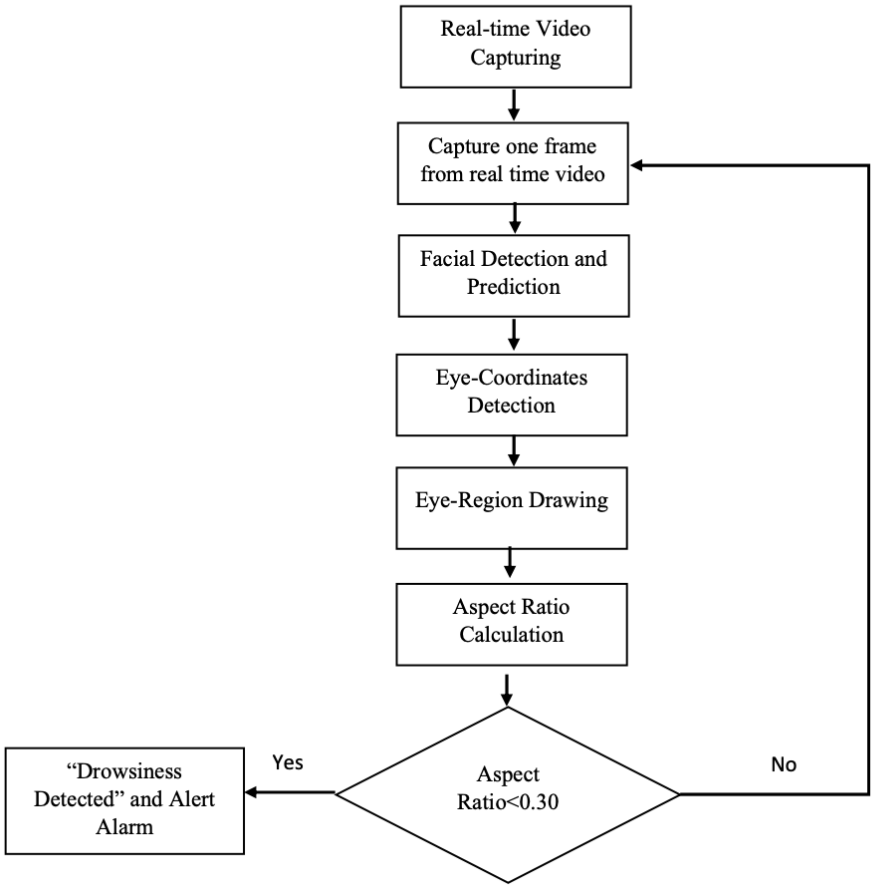


1. Pulse Rate Measuring system



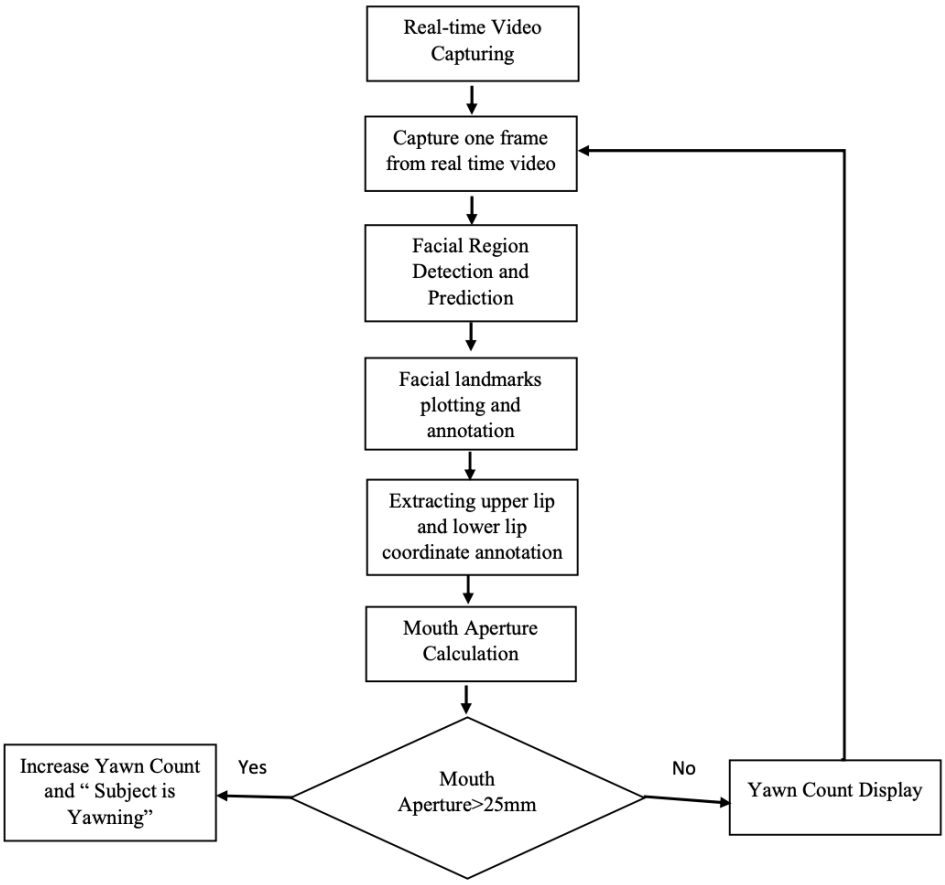
The pulse sensor is interfaced with Node MCU microcontroller and Arduino IDE is used for coding part and connecting the Node MCU with ThingSpeak as a part of central monitoring system and is uploaded back to the microcontroller. As soon as the pulse sensor senses the pulse, the data is sent to ThingSpeak and the reading is received from ThingSpeak and plotted in the Serial plotter of Arduino IDE. When the pulse rate comes below 60, the buzzer will turn on alerting the driver to wake up.

2. Eye Aspect Ratio



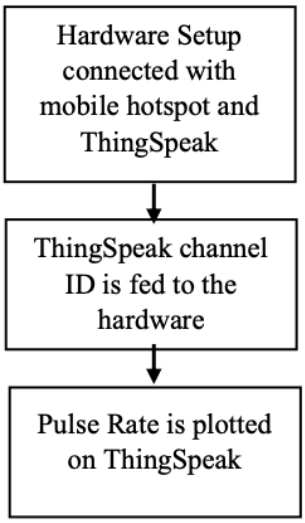
Eye drowsiness detection system detects the aspect ratio of the driver's eye in real time with the help of a camera, sitting in front of the system. The eye bounding coordinates are extracted from the detected area. These coordinates are used to calculate the aspect ratio of the eye. The aspect ratio is then compared to a particular threshold value(0.25), below which the driver is detected as drowsy and is alerted by ringing an alarm, else the process is continued again. When the aspect ratio of the eye comes below 0.25 the buzzer will turn on alerting the driver.

3. **Yawn detection System**



The yawn detection system also detects the yawn count of the driver in real time, with the help of the camera in front of which the driver will be sitting. The midpoint annotation of upper and lower lips are determined and their Euclidean distance is calculated. If the Euclidean distance is above a particular threshold(25 mm) the " Subject is Yawning" is printed and yawn count is increased and displayed else the process is again repeated. When the yawn count is above 5, the buzzer will turn on alerting the driver.

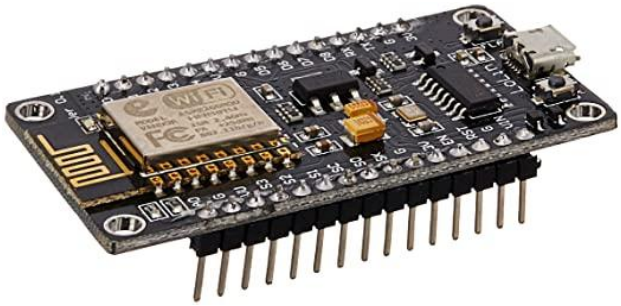
4. Central Monitoring System



The main objective of this system is that a single control room with this system installed alone will be able to monitor the pulse rate, eye aspect ratio and yawn detection and take further actions if signs of drowsiness are noticed. The central monitoring system is connected to the internet. The real time data is then fetched from the hardware to this window and plotted in real time producing the pulse rate graph, yawn detection graph and eye aspect ratio in the Central Monitoring System i.e. ThingSpeak .

VI. Components Used

A. Node MCU

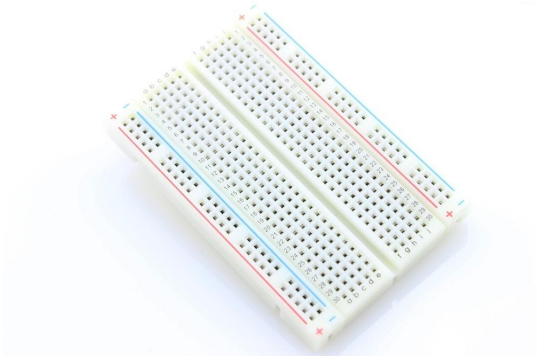


B. Pulse Sensor

C. Buzzer

D. Jumper Wires

E. Bread Board



VII. Analysis

1. Pulse Rate Measuring System

a. Specifications of pulse rate sensor

- Biometric Pulse Rate sensor
- Plug and Play type sensor
- Operating Voltage: +5V or +3.3V
- Current Consumption: 4mA
- Inbuilt Amplification and Noise cancellation circuit.
- Diameter: 0.625”
- Thickness: 0.125” Thick

b. Specifications of Node MCU

- Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Digital I/O Pins (DIO): 16
- Analog Input Pins (ADC): 1
- UARTs: 1
- SPIs: 1
- I2Cs: 1
- Flash Memory: 4 MB

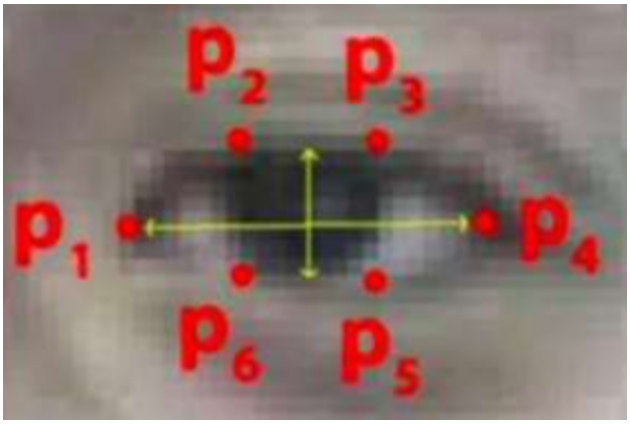
- SRAM: 64 KB
- Clock Speed: 80 MHz
- USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
- PCB Antenna
- Small Sized module to fit smartly inside your IoT projects

c. Functioning

The pulse sensor has a light which helps in measuring the pulse rate. When we place a finger on the light, the light will change according to the blood present in the capillary blood vessels of the finger. During a heartbeat the volume of blood inside the vessels will be high, this will affect the reflection of the light recorded.

The light reflected will be less, when heartbeat occurs as compared to when there is no heartbeat. This variation in the reflection of light recorded by the pulse sensor helps to record the pulse rate. This pulse can be then conditioned to measure heartbeat and then programmed accordingly to read as heartbeat count.

2. Eye Aspect Ratio



In the figure given above coordinates P1,P4 are the extreme ends of the eye and P2,P3,P5,P6 are the intermediate points of the eye. When all these 6 coordinates are found, then the aspect ratio of the eye can be calculated using the following formula :

$$Eye\ aspect\ ratio\ =\ (\frac{|P2-P6|+|P3-P5|}{2*|P1-P4|})$$

The eye aspect ratio is monitored continuously and whenever the ratio goes below 0.25, it indicates that the driver is drowsy.

Face Detector : A face detector is defined using the dlib module. The library uses a pre-trained face detector, which is based on a modification to the histogram of oriented gradients and uses linear SVM (support vector machine) method for object detection. It makes use of the theory that the eye area , mouth area and edges of the face are comparatively darker than the other face areas. These are called Haar features and the haar- cascade file contains resemblance of these features. Hence comparison with those can perfectly predict the eye and mouth area.

Face Predictor: A predictor is defined using the dlib library again where a dat file with facial landmark features are passed for comparison and detection.

Eye coordinate detector : The face utils class is then used to detect the extreme two coordinates of the eye.

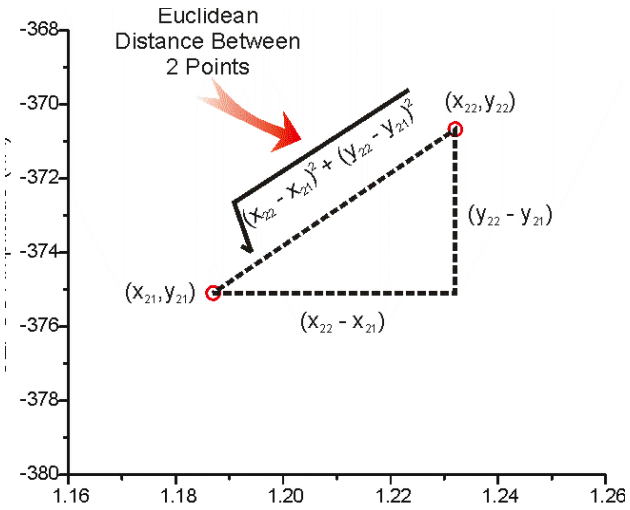
Drawpoints Function : The coordinates of the eye is passed to the following function and it plots the coordinate on eye lids using cv2 library.

Capturing the face : The webcam reads the face and divides it into frames which are then passed to the drowsiness detector function and also displayed on the screen using cv2 library.

3. Yawn Detection System

The coordinates for the lips are detected the same as is done on the eye aspect ratio detection system mentioned above. Once the coordinates are recorded the face coordinate points are plotted all over the face determining the mouth area, eye area and overall face edges.

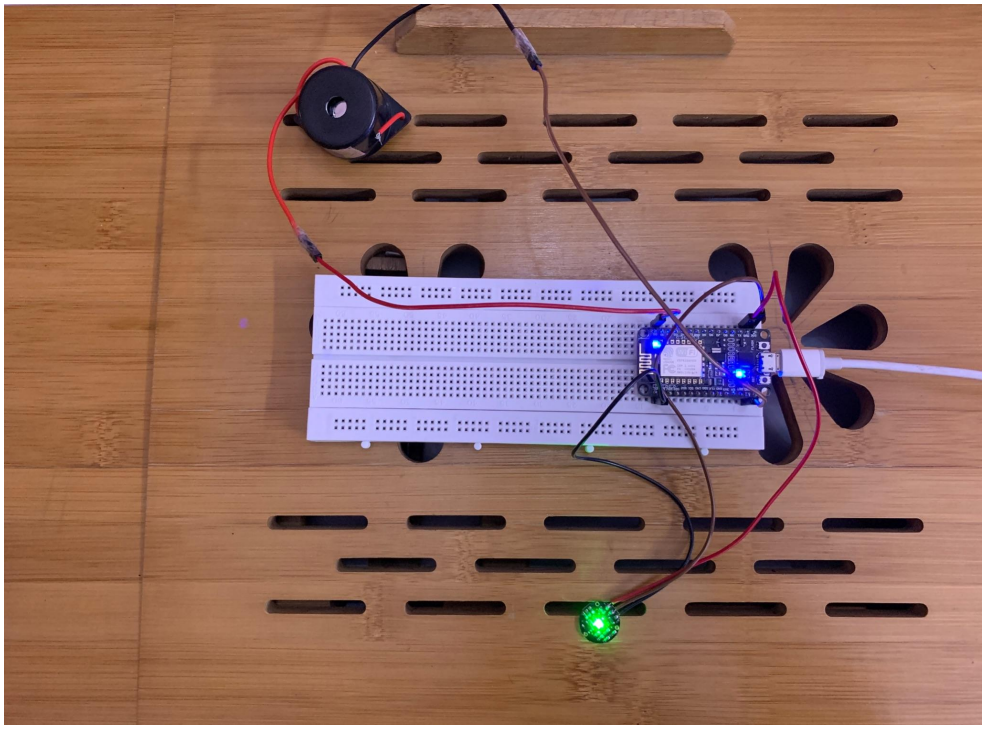
From the landmark coordinates detected, The top lip coordinates are separated and the mean of them i.e. the midpoint coordinate of the upper lip is detected. Similarly, the midpoint of the lower lip is also detected. The distance between these points are calculated using the Euclidean distance formula.

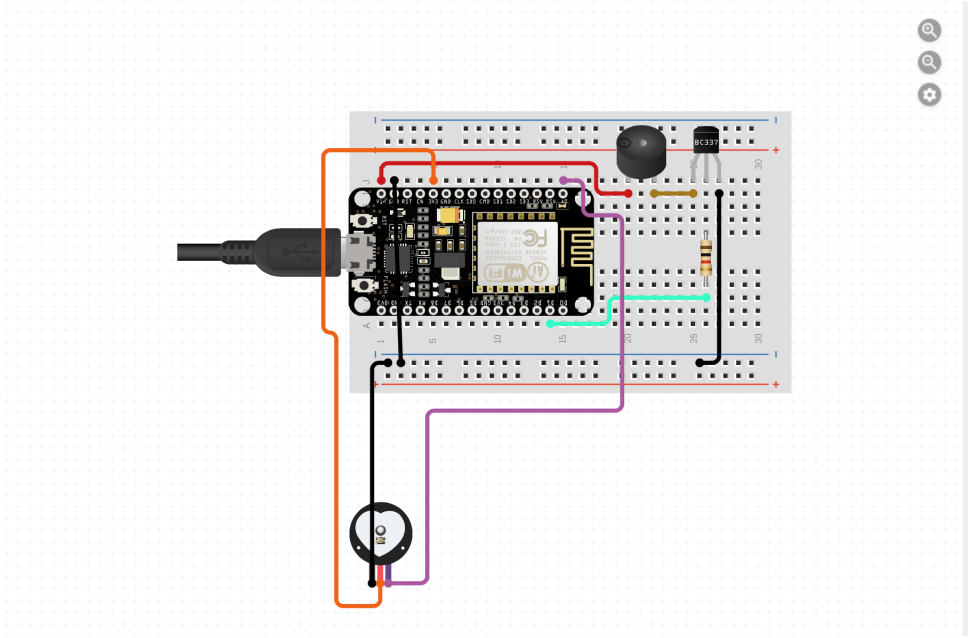


4. Central Monitoring System

Accidents due to driver drowsiness specifically manifest for vehicles wearing goods for a selected employer. The relevant monitoring system is a laptop that is connected via iot and wifi module(in-built in Node MCU) to the pulse fee tracking gadget(via thingspeak software program) and plots the pulse values of the motive force recorded at some point of driving. In this way, the employer can maintain a track if any of the drivers with their goods are drowsy or worn-out and may prevent accidents.

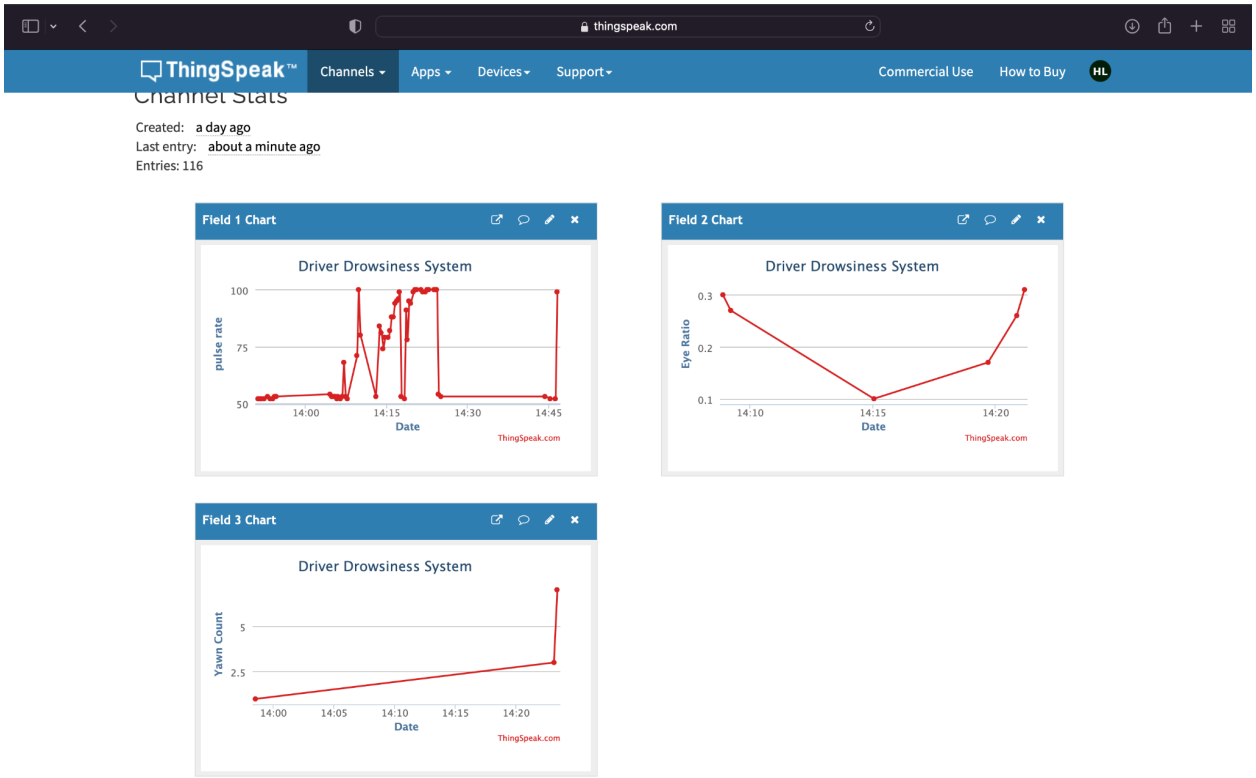
VIII. Outcomes and Screenshots of Circuit



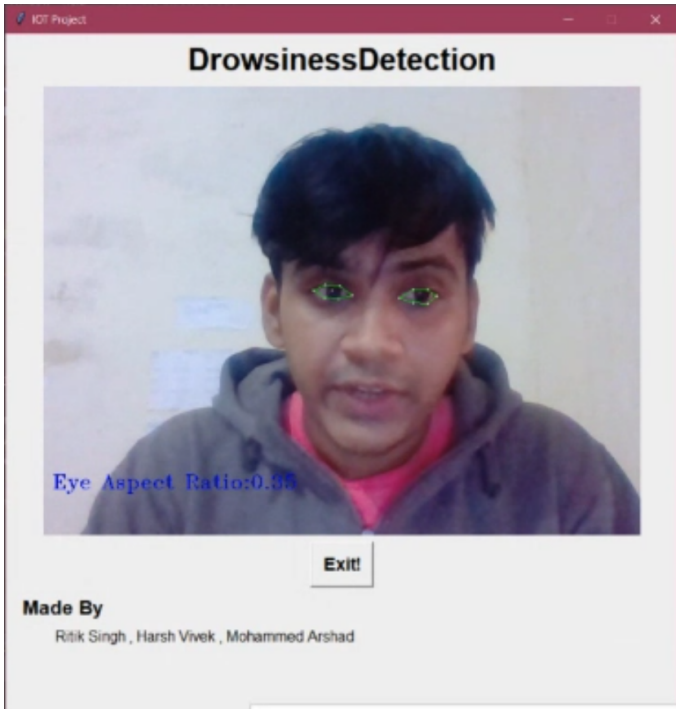


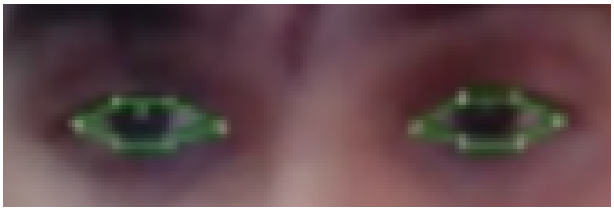
A. Output from Arduino IDE

B. ThingSpeak Data



C. Open CV





IX. Conclusion and Future Work

The proposed system can detect the drowsiness very accurately since three different systems are combined to design our project. Tendency of eye closing and yawning are both considered as signs of drowsiness and in any such case this system perfectly detects and alarms the driver about this condition. The pulse rate measuring and central monitoring system both keeps a check on users pulse rate in real time and thus indicates before the driver can fall asleep. This system will be a very important device to reduce road accidents and death rate due to this.

This system can be made even faster and more efficient by using GSM/GPRS system instead of Wifi. There can be further modifications made in the proposed system to make it efficient and implement it in real time.

X. References

[1] Wanghua Deng, Ruoxue Wu, “Real-Time Driver-Drowsiness Detection System Using Facial Features”, IEEE, Volume-7, August 2019

[2] V B Navya Kiran, Raksha R, Anisoor Rahman, Varsha K N, Dr. Nagamani N P, “Driver Drowsiness Detection”, IJERT, Volume-8 Issue-15, July 2020

[3] Mkhuseli Ngxande, Jules-Raymond Tapamo, Michael Burke, “Driver drowsiness detection using behavioral measures and machine learning techniques: A review of state-of-art techniques”, IEEE, November 2017

XI. Appendix

A. Code for Pulse Rate Sensor using Node MCU

```
#include <ESP8266WiFi.h>;
#include <WiFiClient.h>;
#include <ThingSpeak.h>;

const char* ssid = "Wifi Name"; // Network SSID
const char* password = "Wifi Password";
int val;
int PulseSensorpin = A0;
int buzzer = D0;
float aConst = 2.25E-02;
float bConst = 100.0;
float cConst = 0.0;
WiFiClient client;
unsigned long myChannelNumber = 1587955 ;
unsigned int dataFieldOne = 1;
unsigned int dataFieldTwo = 2;
unsigned int dataFieldThree = 3;


const char * myWriteAPIKey = "ZBJBCPZZJV2LC10L";
const char * myReadAPIKey = "LZRTHV4RUAHCIXN6";


void setup() {

  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(9600);
  delay(10);
```

```
//  pinMode(buzzer, OUTPUT);

   // Connect to WiFi network
  WiFi.begin(ssid, password);
  ThingSpeak.begin(client);

}

void loop() {

    digitalWrite(LED_BUILTIN, LOW);
  delay(1000);                      // Wait for a second
  digitalWrite(LED_BUILTIN, HIGH);
  delay(2000);

  val = analogRead(PulseSensorpin);
  Serial.println("Pulse Sensorvalue= ");
  Serial.println(val*100/1023);
  delay(400);
  ThingSpeak.writeField(myChannelNumber, 1,val*100/1023,
myWriteAPIKey); //Update in ThingSpeak

  aConst = readTSData( myChannelNumber, dataFieldOne );
  Serial.println("Pulse Rate: "+ String(aConst));

  bConst = readTSData( myChannelNumber, dataFieldTwo );
  Serial.println("Eye Ratio: "+ String(bConst));

  cConst = readTSData( myChannelNumber, dataFieldThree );
  Serial.println("Yawn Count: "+ String(cConst));
```

```
  if(aConst<55){
   tone(buzzer,450);
   delay(100);
   noTone(buzzer);
   delay(100);


  }

  else if(bConst<0.25){
   tone(buzzer,450);
   delay(100);
   noTone(buzzer);
   delay(100);


  }

  else if(cConst>=5){
   tone(buzzer,450);
   delay(100);
   noTone(buzzer);
   delay(100);


  }

}

float readTSData( long TSChannel, unsigned int TSField ) {
```

```
  float data =  ThingSpeak.readFloatField( TSChannel, TSField,
myReadAPIKey );
  Serial.println( " Read from ThingSpeak: " + String( data, 9 ) );
  return data;

}
```

B. Code for Eye Aspect Ratio

```
from scipy.spatial import distance
from PIL import Image, ImageTk
from imutils import face_utils
import tkinter as tk
import numpy as np
import dlib
import PIL
import cv2
import threading
import os
import urllib.request
import requests

def postdata(counter):
    URL = 'https://api.thingspeak.com/update?api_key='
    KEY = 'ZBJBCPZZJV2LC10L'
    HEADER = '&field2={}'.format(counter)
    NEW_URL = URL+KEY+HEADER
    v = urllib.request.urlopen(NEW_URL)
    print(v)

# Aspect Ratio Calculation
def eye_aspect_ratio(eye):
```

```
    a = distance.euclidean(eye[1], eye[5])
    b = distance.euclidean(eye[2], eye[4])
    c = distance.euclidean(eye[0], eye[3])
    return (a + b) / (2 * c)


# Drawing eye points
def drawPoints(eye, frame):
    for cent in eye:
        cv2.circle(frame, tuple(cent), 1, (255, 255, 255), 1)


# Detection of drowsiness
def drowsy_detection(frame):
    global COUNTER
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = detector(gray, 0)
    for face in faces:
        shape = predictor(gray, face)
        shape = face_utils.shape_to_np(shape)
        leftEye = shape[lStart:lEnd]
        rightEye = shape[rStart:rEnd]

        drawPoints(leftEye, frame)
        drawPoints(rightEye, frame)
        l_eye = eye_aspect_ratio(leftEye)
        r_eye = eye_aspect_ratio(rightEye)
        eye = (l_eye + r_eye) / 2
        cv2.drawContours(frame, [leftEye], -1, (0, 255, 0), 1)
        cv2.drawContours(frame, [rightEye], -1, (0, 255, 0), 1)
        # Threshold Calculation
        if eye < EYE_THRESHOLD:
```

```
            COUNTER += 1
            if COUNTER >= EYE_FRAMES:
                eye_str = str(eye)
                cv2.putText(frame, "ALERT: Drowsiness detected.", (30,
50),cv2.FONT_HERSHEY_TRIPLEX, 1.0, (0, 0, 222), 2)
                cv2.putText(frame, ("Eye Aspect Ratio:" + eye_str[:4]), (10, h -
50), cv2.FONT_HERSHEY_TRIPLEX, 0.7,(222, 0, 0), 1)
                threading.Thread(target=postdata, args=(eye_str[:4],)).start()
        else:
            COUNTER = 0
            eye_str = str(eye)
            print(eye_str[:4])
            # URL = 'https://api.thingspeak.com/update?api_key='
            # KEY = 'RHOAD5VJ52XOY9TX'
            # HEADER = '&field2={}'.format(eye_str[:4])
            # NEW_URL = URL + KEY + HEADER
            # v = urllib.request.urlopen(NEW_URL)
            # print(v)
            cv2.putText(frame, ("Eye Aspect Ratio:" + eye_str[:4]), (10, h -
50),cv2.FONT_HERSHEY_TRIPLEX, 0.7, (222, 0, 0), 1)
    return frame


# Capturing and displaying the picture
def video_loop():
    ret, frame = cam.read()
    if ret:
        frame = drowsy_detection(frame)
        # print(frame)
        cv2image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGBA)
        current_image = Image.fromarray(cv2image)
        imgtk = ImageTk.PhotoImage(image=current_image)
```

```
        panel.imgtk = imgtk
        panel.config(image=imgtk)
        root.after(30, video_loop)


# Destroying the windows
def destructor():
    print("[INFO] closing...")
    root.destroy()
    cam.release()
    cv2.destroyAllWindows()


# The main function
if __name__ == "__main__":
    EYE_THRESHOLD = 0.25
    EYE_FRAMES = 15
    COUNTER = 0

    cam = cv2.VideoCapture(0, cv2.CAP_DSHOW)
    detector = dlib.get_frontal_face_detector()
    predictor =
dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
    (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS['left_eye']
    (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS['right_eye']
    h = 480
    w = 640
    cam.set(3, h)
    cam.set(4, w)
    current_image = None
    print("[INFO] Starting...")
    root = tk.Tk()
```

```
    root.geometry("720x740")
    root.resizable(False, False)
    root.title("IOT Project")
    root.protocol("WM_DELETE_WINDOW", destructor)
    title = tk.Label(root, font="Helvetica 25 bold",
text="DrowsinessDetection").pack(anchor='center', padx='5',pady='5')
    panel = tk.Label(root)
    panel.pack(anchor='center')
    root.config(cursor="arrow")
    btn = tk.Button(root, text="Exit!", font="Helvetica 14 bold",
command=destructor)
    btn.pack(anchor='center', ipadx=5, ipady=5, pady=5)
    team = tk.Label(root, font="Helvetica 16 bold", text="Made
By").pack(anchor='w',ipadx=5, ipady=2, padx=10)
    mate1 = tk.Label(root, font="Helvetica 12 ", text="Ritik Singh , Harsh
Vivek , Mohammed Arshad").pack(anchor='w',padx=50, ipady=1)
    video_loop()
    root.mainloop()
```

C. Code for Yawn Detection

```
import cv2
import dlib
import numpy as np
import urllib.request
import requests
import threading


#Detector and Predictor
PREDICTOR_PATH = "shape_predictor_68_face_landmarks.dat"
predictor = dlib.shape_predictor(PREDICTOR_PATH)
detector = dlib.get_frontal_face_detector()
```

```
def postdata(yawns):
    URL = 'https://api.thingspeak.com/update?api_key='
    KEY = 'ZBJBCPZZJV2LC10L'
    HEADER = '&field3={}'.format(yawns)
    NEW_URL = URL + KEY + HEADER
    v = urllib.request.urlopen(NEW_URL)
    print(v)
    print(yawns)

#Facial Landmarks drawing
def get_landmarks(im):
    rects = detector(im, 1)
    if len(rects) > 1:
      return "error"
    if len(rects) == 0:
      return "error"
    return np.matrix([[p.x, p.y] for p in predictor(im, rects[0]).parts()])

def annotate_landmarks(im, landmarks):
    im = im.copy()
    for idx, point in enumerate(landmarks):
        pos = (point[0, 0], point[0, 1])
        cv2.putText(im, str(idx), pos,
fontFace=cv2.FONT_HERSHEY_SCRIPT_SIMPLEX, fontScale=0.4,
color=(0, 0, 255))
        cv2.circle(im, pos, 3, color=(0, 255, 255))
    return im

def top_lip(landmarks):
```

```
    top_lip_pts = []
    for i in range(50,53):
        top_lip_pts.append(landmarks[i])
    for i in range(61,64):
        top_lip_pts.append(landmarks[i])
    top_lip_all_pts = np.squeeze(np.asarray(top_lip_pts))
    top_lip_mean = np.mean(top_lip_pts, axis=0)
    return int(top_lip_mean[:,1])

def bottom_lip(landmarks):
    bottom_lip_pts = []
    for i in range(65,68):
        bottom_lip_pts.append(landmarks[i])
    for i in range(56, 59):
        bottom_lip_pts.append(landmarks[i])
    bottom_lip_all_pts = np.squeeze(np.asarray(bottom_lip_pts))
    bottom_lip_mean = np.mean(bottom_lip_pts, axis=0)
    return int(bottom_lip_mean[:, 1])


# The mouth openness measure
def mouth_open(image):
    landmarks = get_landmarks(image)

    if landmarks == "error":
        return image, 0
    image_with_landmarks = annotate_landmarks(image, landmarks)
    top_lip_center = top_lip(landmarks)
    bottom_lip_center = bottom_lip(landmarks)
    lip_distance = abs(top_lip_center - bottom_lip_center)
    return image_with_landmarks, lip_distance
```

```
#Video capture and main function
cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
yawns = 0
yawn_status = False

while True:
    ret, frame = cap.read()
    image_landmarks, lip_distance = mouth_open(frame)

    prev_yawn_status = yawn_status

    if lip_distance > 25:
        yawn_status = True

        cv2.putText(frame, "Subject is Yawning", (50,
450),cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 2)
        output_text = " Yawn Count: " + str(yawns + 1)
        # URL = 'https://api.thingspeak.com/update?api_key='
        # KEY = 'RHOAD5VJ52XOY9TX'
        # HEADER = '&field3={}'.format(yawns+1)
        # NEW_URL = URL + KEY + HEADER
        # v = urllib.request.urlopen(NEW_URL)
        # print(v)
        # print(yawns+1)

        cv2.putText(frame, output_text, (50,
50),cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 127), 2)
    else:
        yawn_status = False

    if prev_yawn_status == True and yawn_status == False:
        yawns += 1
```

```
        if yawns >= 5:
            threading.Thread(target=postdata,args=(yawns,)).start()


    cv2.imshow('Live Landmarks', image_landmarks)
    cv2.imshow('Yawn Detection', frame)

    cv2.imshow('Live Landmarks', image_landmarks)
    cv2.imshow('Yawn Detection', frame)

    if cv2.waitKey(1) == 13 :
        break

cap.release()
cv2.destroyAllWindows()
```