DriveSafe360: Intelligent 360-degree Safety for Smart Cars using Deep Learning & IoT

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology
in
Computer Science and Engineering

by

KOTHA V V S AAKASH

19BCE0186

&

KOTHAMASU KARTHIK

19BCI0141

Under the guidance of

Dr. Gopichand G

School of Computer Science and Engineering

VIT, Vellore

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

May 2023

DECLARATION

I hereby declare that the thesis entitled “DriveSafe360: Intelligent 360-degree Safety for Smart Cars using Deep Learning & IoT” submitted by me, for the award of the degree of Bachelor of Technology in Computer Science and Engineering to VIT is a record of bonafide work carried out by me under the supervision of Dr. Gopichand G, School of Computer Science and Engineering, VIT.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree ordiploma in this institute or any other institute or university.

Place: Vellore

Date: 20/05/2023

Signature of the Candidate

[illegible]

<u>CERTIFICATE</u>

This is to certify that the thesis entitled "DriveSafe360: Intelligent 360-degree Safety for Smart Cars using Deep Learning & IoT" submitted by KOTHA V V S AAKASH **&** 19BCE0186, School of Computer Science and Engineering, VIT, for the award of the degree of Bachelor of Technology in Computer Science and Engineering, is a record of bonafide work carried out by him/her under my supervision during the period, 01.07.2022 to 30.04.2023, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute orany other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion, meets the necessary standards for submission.

Place: Vellore

Date: 20/05/2023

Signature of the Guide

Internal Examiner **External Examiner**

Dr. VAIRAMUTHU S

Computer Science and Engineering

ACKNOWLEDGEMENTS

It is my pleasure to express with a deep sense of gratitude to Dr. Gopichand G, Assistant Professor Sr. Grade 1, SCOPE School, Vellore Institute of Technology, for his constant guidance, continual encouragement, and understanding; more than all, he taught me patience in my endeavor. My association with him is not confined to academics only, but it is a great opportunity on my part to work with an intellectual and expert in the fields of Deep Learning and IoT.

I would like to express my gratitude to Dr. G. Viswanathan, Chancellor, Vellore Institute of Technology, Vellore, Mr. Sankar Viswanathan, Dr. Sekar Viswanathan, Mr. G V Selvam, Vice – Presidents, Vellore Institute of Technology, Vellore, and Dr. Ramesh Babu K, Dean, School of Computer Science & Engineering (SCOPE), for providing an environment to work in and for his inspiration during the tenure of the course.

In a Jubilant mood, I express ingeniously my whole-hearted thanks to Dr. Vairamuthu S, HoD, all teaching staff and members working as limbs of our university for their not-self-centered enthusiasm coupled with timely encouragement showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my course study successfully. I would like to thank my parents for their support. It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. Last but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.

KOTHA V V S AAKASH

Executive Summary

This project aims to develop DriveSafe360, an intelligent 360-degree safety system for smart cars that leverages deep learning and IoT technologies. The system's primary objective is to prevent accidents by detecting driver Fatigue and alcohol consumption in drivers. DriveSafe360 will be seamlessly integrated into the car's onboard computer, constantly monitoring driver behavior and issuing alerts when necessary to ensure driver safety. This model will undergo training on a large dataset containing images of drowsy and non-drowsy drivers. By leveraging deep learning and ensemble learning, we aim to provide an effective and precise solution for identifying driver fatigue promptly. We also propose an architecture that uses an MQ-3 sensor connected to an Arduino UNO-3 to detect the level of alcohol in a person's breath. The system triggers a piezo buzzer and an LED if a high level of alcohol is detected. It also displays a message on an LCD screen, giving the driver an option to enable the emergency mode if necessary.

CONTENTS

List of Figures

List of Tables

List of Abbreviations

YOLO	You only look once
HOG	Histogram of oriented gradients
IoT	Internet of Things
OpenCV	Open-Source Computer Vision Library
MQ-3	Grove - Gas Sensor
EAR	Eye Aspect Ratio
MAR	Mouth Aspect Ratio

Symbols and Notations

Ω	Ohm

1. INTRODUCTION

1.1. Theoretical Background

The increasing number of road accidents has become a key concern for society globally and in India. According to the World Health Organization, road traffic accidents are the leading cause of death among young people aged 15-29 and account for 1.35 million deaths each year worldwide. In India, according to the Ministry of Road Transport and Highways, in 2019 alone, around 150,000 people lost their lives due to road accidents.

Furthermore, a significant cause of accidents is driver behavior, including driver fatigue and alcohol-impaired driving (accounting for around 50-60% of road accidents in India and globally). Research shows that drowsy driving accounts for 2-3% of all road accidents, while alcohol-impaired driving accounts for nearly 30% of all accidents. To address this pressing concern, the need of the hour is the development of advanced technologies in cars, such as deep learning and IoT, to assist drivers in avoiding accidents.

Considering these alarming statistics, our research aims to address this issue by developing DriveSafe360, an intelligent 360-degree safety system for smart cars that utilize deep learning and IoT technologies. By implementing real-time driver monitoring using a camera (computer vision) and an IoT alcohol sensor, DriveSafe360 detects driver fatigue and alcohol consumption in drivers, preventing accidents before they occur.

The system uses a camera (computer vision) mounted on the vehicle’s dashboard to detect drowsiness to capture real-time images of the driver's face and eyes. These images are then fed into a deep-learning model trained to detect signs of drowsiness, such as drooping eyelids or a lack of eye movement. If the model detects signs of drowsiness, it triggers an alert to the driver, reminding them to take a break or pull over.

To detect alcohol consumption, the system utilizes an IoT alcohol sensor that is integrated into the vehicle's cabin. The sensor continuously monitors the driver's breath for the presence of alcohol. If the sensor detects a high level of alcohol, it triggers an alert to the driver, warning them against driving under the influence.

In conclusion, DriveSafe360 is an innovative solution that utilizes deep learning and IoT technologies to detect driver fatigue and alcohol consumption in drivers to prevent accidents before they occur, making the roads safer for all. It is designed to be easy to install and use, providing a comprehensive and reliable system for ensuring driver safety.

1.2. Motivation

The project "DriveSafe360: Intelligent 360-degree Safety for Smart Cars using Deep Learning & IoT" is driven by the compelling motivation to revolutionize road safety and protect lives in the era of smart cars. With the rise of advanced technologies like deep learning and IoT, it is essential to harness their potential to create a comprehensive safety solution. The primary objective is to reduce accidents, injuries, and fatalities by ensuring that smart cars are equipped with an intelligent safety system that operates on a 360-degree basis. By incorporating deep learning algorithms and IoT connectivity, the project aims to provide real-time monitoring, risk assessment, and proactive hazard detection capabilities to prevent accidents before they occur. The motivation extends beyond just enhancing safety; it also strives to enhance the overall driving experience for smart car owners, empowering them with confidence and peace of mind on the roads. By pushing the boundaries of innovation and technological advancement, DriveSafe360 sets out to shape the future of automotive technology, contributing to a safer, more efficient, and more connected transportation ecosystem. Through collaboration, industry influence, and a human-centered approach, this project aims to establish itself as the benchmark for intelligent safety in smart cars, influencing industry standards and paving the way for a safer future for all road users.

1.3. Aim of the Proposed Work

The increasing number of road accidents in India has become a pressing concern for society, with drowsiness and alcohol-impaired driving being significant contributing factors. Smart cars equipped with advanced technologies such as deep learning and IoT have the potential to significantly reduce the number of accidents on the road by detecting driver fatigue and alcohol-impaired driving in real time and monitoring the speed limit. DriveSafe360 is a proposed intelligent 360-degree safety system for smart cars that utilize deep learning and IoT to prevent accidents by detecting driver fatigue and alcohol-impaired driving in real-time.

The system uses advanced deep learning algorithms to analyze data from sensors such as cameras mounted inside the car. These sensors constantly capture data from the car's interior and location and feed it to deep learning algorithms, which then analyze the data to detect potential hazards such as drowsiness. The system also uses IoT to connect the car with sensors, such as a camera and Alcohol sensor, to gather additional real-time information about the car's driver.

DriveSafe360 has the capability to detect driver fatigue and alcohol-impaired driving in real-time by monitoring the driver's behavior and physiological signs. When the system detects drowsiness or alcohol-impaired driving, it alerts the driver through visual and audio warnings. It can also take autonomous actions such as slowing down or even stopping the car if necessary.

Fig-1: Project Logo

1.4. Objective of the Proposed Work

The Objective of our research is to develop DriveSafe360, an intelligent 360-degree safety system for smart cars that utilize deep learning and IoT technologies to prevent accidents by detecting driver fatigue and alcohol consumption in drivers. The system is designed to be easy to install and use, providing a comprehensive and reliable solution for ensuring driver safety. DriveSafe360 will be integrated into the car's onboard computer, constantly monitoring the driver's behavior, and alerting them when necessary to prevent accidents.

The main step in achieving this objective is to develop a deep-learning model that detects driver fatigue using real-time images of the driver's eyes captured by a camera. This model will be trained on a large dataset of photos of drowsy and non-drowsy drivers and will be able to detect drowsiness in real-time accurately. Additionally, the system will use IoT technology to detect the presence of alcohol on the driver's breath.

2. LITERATURE SURVEY

S.No	Details	Approach Used	Research Gaps Identified
1.	**Title:** IoV Road Safety: Vehicle Speed Limiting System [1] **Authors:** Abdelsalam Mohamed, Bonny Talal **Publisher:** IEEE **Year:** 2019	The proposed system in this research paper uses a transmitter unit fixed on speed signs and a receiver unit installed in the car. The transmitter unit sends a message signal containing the speed limit of the road to the receiver unit, which then compares the car's current speed to the speed limit and adjusts the car's speed if necessary. The driver has complete control over the vehicle except for increasing the speed above the set speed limit. The system can also convey other types of information, such as warning messages in bad weather conditions, suggested routes to avoid traffic congestion, and emergency messages in case of car accidents. The transmitter units are connected to a central server which facilitates to send and receive any information needed.	Firstly, while the transmitter-receiver units enable speed limit enforcement, there is no mention of how the accuracy and reliability of the transmitted speed limit information are ensured, which could potentially lead to incorrect speed adjustments.
2.	**Title:** Safe Drive: An Automatic Engine Locking System to Prevent Drunken Driving[2] **Authors:** Sharanabasappa Sayed Farooq Soundarya, V.N Rao, Vikram S Chandraprabha **Publisher:** IEEE **Year:** 2018	The proposed system uses an IoT-based MQ-3 gas sensor to detect alcohol, an Arduino Uno R3 microcontroller board, a Raspberry Pi 3, and DC motors to simulate car engines. If the alcohol level detected by the sensor exceeds the threshold level, the power to the ignition of the engine will be cut off, and a message will be sent to a remote server. The remote server will then send a notification to the driver or owner of the vehicle asking them to take a sobriety test through an Android application. If the user passes the test, they can unlock the engine and drive the car. If the user fails the test, the engine will remain off, and the driver's location and details will be sent to the server to be stored in a database.	The survey does not provide sufficient information about the accuracy and reliability of the MQ-3 gas sensor in detecting alcohol levels, which is crucial for ensuring the effectiveness of the system.

S.No	Details	Approach Used	Research Gaps Identified
3.	**Title:** An Integrated Framework for Driver Drowsiness Detection and Alcohol Intoxication using Machine Learning [3]. **Authors:** Varghese, R.R. Jacob, P.M. Jacob, J. Babu, M.N. Ravikanth, R. George, S.M. **Publisher:** IEEE **Year:** 2021	The proposed system uses an alcohol sensor to detect traces of alcohol in the driver's breath. If no alcohol is detected, it activates a drowsiness detection system using an infrared night vision camera placed on the vehicle’s dashboard. The system captures video and extracts 2-D images using a Histogram of Oriented Gradients (HOG) algorithm and linear Support Vector Machine (SVM) for classification to detect the driver's face. It then uses 68 facial landmark points to compute decision-making parameters such as Eye aspect ratio (EAR) and Mouth opening ratio (MAR) to detect drowsiness. The system uses a combination of predetermined threshold values and the SVM classifier to detect drowsiness and caution the driver using an alarm. The alcohol detection and drowsiness detection systems are integrated using Arduino UNO and Raspberry Pi 3. The system architecture is divided into several modules: data acquisition and pre-processing, face detection, and facial landmark marking.	There is a lack of discussion regarding the real-time response and effectiveness of the alarm system in alerting and preventing potential accidents caused by drowsiness.
4.	**Title:** A Practical Implementation of Driver Drowsiness Detection Using Facial Landmark. [4] **Authors:** Siwach, Meena Mann, Suman Gupta, Deepa **Publisher:** IEEE **Year:** 2022	This paper proposed using an integral image and an AdaBoost-based learning algorithm for efficient feature computation and selection in facial detection. They also used an ensemble of regression trees to evaluate facial landmark positions in real time with high-quality predictions, similar to the approach used in the dlib library. The research extends this method to live video streams, using technologies like Golang and SASS and incorporates an encoding process to build a user-friendly application connected to a server. The goal is to detect facial landmarks precisely enough to estimate the eye-opening level.	A research gap in this approach is that the algorithm is not robust to different types of people, for example, people with glasses, which can affect facial landmarks detection. Additionally, the approach relies on the AdaBoost-based learning algorithm to select a small number of critical features, which could lead to a loss of important information if not all features are considered.

S.No	Details	Approach Used	Research Gaps Identified
5.	**Title:** Alcohol Sensing Alert with Engine Locking [5]. **Authors:** Kanishka Jose, Sangeeth M , Raj Arya S Sarath M , Vidya Surendran **Publisher:** JETIR. **Year:** 2021	This research paper proposes a system that uses an alcohol sensor, Arduino Uno microcontroller, LCD, buzzer, relay, and LED to detect the presence of alcohol in a driver's breath and alert the driver, and an engine locking mechanism. The system is programmed to work synchronously, with the reading displayed on the LCD board. The system uses an MQ3 sensor, a piezoelectric crystal-type buzzer, and a DC gear motor. When alcohol is detected, the system will activate an alarm through the buzzer and show the presence of alcohol on the LCD panel, and the engine will not start. In the absence of alcohol, the engine will start, and the buzzer will remain silent.	The paper needs to mention how the system handles false positives, which could be a significant drawback if it locks the engine when it is not supposed to.
6.	**Title:** Driver Drowsiness Detection System Based on Visual Features.[6] **Authors:** Fouzia Roopalakshmi, R. Rathod, J.A. Shetty, A.S. Supriya, K. **Publisher:** IEEE. **Year:** 2018	The proposed system aims to detect and alert drowsy drivers by using a camera mounted in front of the driver to continuously capture real-time video, analyze the driver's eye status, and issue a warning signal through a buzzer and vibration on positive detection. The system uses OpenCV and a Raspberry Pi to process the video and determine if the driver's eyes are closed or drowsy. The system also uses a shape predictor to predict the state of the driver's eye and alerts the driver if drowsiness is detected. The system is evaluated using a Raspberry Pi module 3 as a pre-processor and a USB-powered web camera to detect the driver's eye.	The survey lacks sufficient information regarding the accuracy and reliability of the eye status analysis algorithm in determining drowsiness accurately. The performance metrics and evaluation results are not provided, making it challenging to assess the system's effectiveness.

S.No	Details	Approach Used	Research Gaps Identified
7.	**Title:** Embedded Vehicle Speed Control and Over-Speed Violation Alert Using IoT [7] **Authors:** Reddy K., A. Patel, S. Bharath, K.P. Kumar M., R. **Publisher:** IEEE. **Year:** 2019	The proposed method for detecting speed signs involves several morphological operations to detect the speed label from a sign board. Pre-processing involves using a camera to capture an image or video of the speed sign board, converting it to grayscale, and using CANNY edge detection. Character recognition is done, and the obtained information is sent to a cloud server using IoT and accessed using an authority search module. The system compares the speed label and the instantaneous speed of the vehicle and sends a warning message if the speed is greater than the threshold.	The proposed method uses a Hall-effect sensor for real-time speed control, which is tedious and may not be practical in real-world scenarios. And the proposed method is that the system may not accurately detect blurry or defocused images, which could lead to errors in identifying the speed label.
8.	**Title:** IoT based Vehicle Over-Speed Detection and Accident Avoidance System [8] **Authors:** T., Hari Chandan Nagaraju,Shamanth Varma, Bharath M., Kiran Kumar S., Manasa V., Mukund K **Publisher:** IEEE. **Year:** 2021	The proposed system is an IoT-based system that uses ultrasonic sensors to collect data and alert the driver. The system includes an Ultrasonic Sensor, Arduino UNO, Potentiometer, CAN Controller, DC Motor, GSM, LCD, and a buzzer. The ultrasonic sensor detects objects or vehicles in front of the car and sends data to the Arduino UNO, which controls the vehicle's speed. The system also includes over-speed detection, which alerts the driver if the car exceeds a specific speed limit. The system aims to prevent collisions by using ultrasonic sensors to measure the distance between the front vehicle and alert the driver with appropriate warning signals. The system uses the CAN protocol and the Arduino UNO to connect the sensors.	This approach is relies on the ultrasonic sensor to detect objects or vehicles in front of the car. These sensors can be affected by weather conditions such as fog or rain, reducing their accuracy. Additionally, ultrasonic sensors may not be able to detect all types of objects or vehicles, particularly those that are low to the ground.

S.No	Details	Approach Used	Research Gaps Identified
9.	**Title:** Monitoring of Road Accidents- A Review [9] **Authors:** Ghosh, Arijit Prasad, Rajendra **Publisher:** IEEE. **Year:** 2018	This paper highlights the issue of road accidents and the need for effective prevention techniques. It specifically addresses the high rate of accidents in fast-growing economies like India. According to the World Health Organization, around 125 million people globally lose their lives in road accidents annually. In India, the estimated GDP loss due to road accidents is $58 billion, second only to Japan's $63 billion. The paper also mentions the high death tolls in countries with high populations and the need for solutions such as congestion charging and traffic information gathering.	The approach outlined in the paper focuses primarily on the statistics and economic impact of road accidents. Additionally, the approach does not consider the social and cultural factors contributing to road accidents. Also, it only focuses on road accidents in India, and the statistics may not be generalizable to other countries.
10.	**Title:** Forecasting of Road Accident in Kerala: A Case Study[10]. **Authors:** Sunny, C.M. Nithya, S. Sinshi, K.S. Vinodini M.D., Lakshmi K.G., A. Anjana, S. Manojkumar **Publisher:** IEEE. **Year:** 2018	In summary, traffic accidents are a significant cause of death and injuries globally, particularly in developing nations like India. The analysis of road accident data is crucial for taking preventive measures. This study used time series analysis to predict road accidents in Kerala, India, from January 1999 to December 2016. The data shows a trend of increasing road accidents and fatalities due to population growth, improved financial status, and an increased number of vehicles. Other contributing factors include human errors, overspeeding, lack of knowledge about rules, and violation of traffic rules. The World Health Organization predicts that traffic fatalities will be the third leading cause of death worldwide by 2020. Kerala has a high road accident rate, and the authorities struggle to address road maintenance and lack of street lights. The data suggest a need for strict measures and laws to reduce road accidents.	One drawback of the above approach is that it focuses solely on the statistics of road accidents in Kerala, India. It does not consider other potential factors that may contribute to the accidents, such as weather conditions or road infrastructure. Additionally, the approach only looks at data from January 1999 to December 2016, which may not accurately reflect current or future trends in road accidents.

3. OVERVIEW OF THE PROPOSED SYSTEM

3.1. Introduction and Related Concept

The project "DriveSafe360: Intelligent 360-degree Safety for Smart Cars using Deep Learning & IoT" aims to enhance road safety by employing advanced technologies such as Deep Learning and IoT. It consists of two modules: Human Fatigue (Drowsiness) detection and Alcohol Detection.

One of the modules in the DriveSafe360 project focuses on Alcohol Detection. For this purpose, the project utilizes a sensor called MQ-3. MQ-3 is a gas sensor that is specifically designed to detect the presence of alcohol vapor in the surrounding environment. It operates on the principle of semiconductor gas sensing, where it detects changes in electrical conductivity upon exposure to alcohol fumes. By measuring these changes, the MQ-3 sensor can identify the presence and concentration of alcohol, which is crucial for alcohol detection in smart cars.

In the Human Fatigue (Drowsiness) detection module, the project employs a combination of three powerful technologies: Dlib, YOLOv8, and cvzone. These tools work collectively to detect drowsiness in real time.

i. Dlib: Dlib is a popular open-source library that provides various machine-learning algorithms and tools. It offers pre-trained models for facial landmark detection, which allows the system to accurately identify key facial features such as eyes, eyebrows, and mouth. By tracking these landmarks, Dlib enables the detection of facial expressions and movements that indicate drowsiness.

ii. YOLOv8: YOLOv8 (You Only Look Once version 8) is an object detection algorithm that uses deep neural networks to identify objects in real time. In the DriveSafe360 project, YOLOv8 is trained on a dataset of closed and open eyes to specifically recognize eye states. It can identify whether the driver's eyes are closed or open, which is a critical factor in detecting drowsiness.

iii. cvzone: cvzone is a computer vision library built on top of OpenCV, which provides additional functionalities and simplifies complex computer vision tasks. It integrates seamlessly with Dlib and YOLOv8, facilitating the combination of their outputs for drowsiness detection. cvzone offers utilities for face detection, facial landmarks, and visualizing the detected objects, making it an essential component for the project.

3.2. Framework, Architecture, or Module for the Proposed System

The system architecture for DriveSafe360 is designed to efficiently integrate advanced technologies such as deep learning and IoT to prevent accidents by detecting drowsiness and alcohol consumption in drivers. The system uses a combination of sensors, including cameras, Breathalyzer sensors, along with computer vision techniques to detect the driver's face, eyes, and gaze, and detect the presence of alcohol. The system is designed to be easy to install and use, providing a comprehensive and reliable solution to ensure driver safety. The system architecture comprises two main modules: Human Fatigue (Drowsiness) detection, and Alcohol Detection.

3.2.1. Human Fatigue Detection Module:

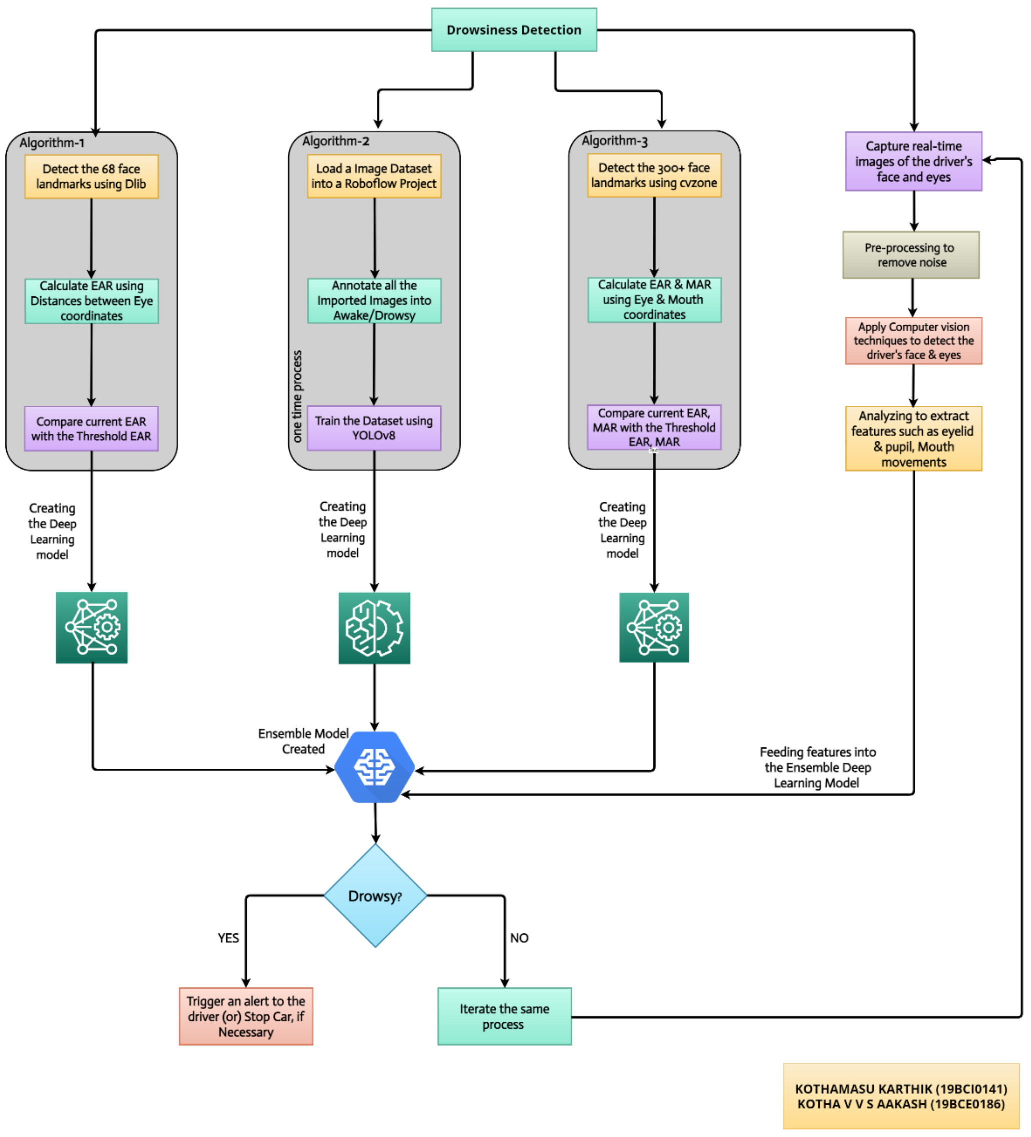

Drowsiness Detection
Algorithm-1
Detect the 68 face landmarks using Dlib
Calculate EAR using Distances between Eye coordinates
Compare current EAR with the Threshold EAR
Algorithm-2
Load a Image Dataset into a Roboflow Project
Annotate all the Imported Images into Awake/Drowsy
Train the Dataset using YOLOv8
one time process
Algorithm-3
Detect the 300+ face landmarks using cvzone
Calculate EAR & MAR using Eye & Mouth coordinates
Compare current EAR, MAR with the Threshold EAR, MAR
Capture real-time images of the driver's face and eyes
Pre-processing to remove noise
Apply Computer vision techniques to detect the driver's face & eyes
Analyzing to extract features such as eyelid & pupil, Mouth movements
Creating the Deep Learning model
Creating the Deep Learning model
Creating the Deep Learning model
Ensemble Model Created
Feeding features into the Ensemble Deep Learning Model
Drowsy?
YES
NO
Trigger an alert to the driver (or) Stop Car, if Necessary
Iterate the same process
KOTHAMASU KARTHIK (19BCI0141)
KOTHA V V S AAKASH (19BCE0186)

Fig-2: System Architecture of Human Fatigue Detection

The system architecture consists of four main components: input, pre-processing, ensemble model, and output.

- **Input:** The input component comprises a camera that captures the driver's face and eyes. The captured images are then passed to the pre-processing component for further analysis.
- **Pre-processing:** The pre-processing component performs several steps to prepare the images for the ensemble model. Firstly, it applies image cropping to isolate the driver's face and eyes. Then, it applies image normalization to reduce illumination variations, followed by image resizing to fit the input dimensions of the ensemble model.
- **Ensemble Model:** The ensemble model combines the outputs of three deep learning models, Dlib, cvzone and YOLOv8, to improve the detection accuracy. The Dlib & cvzone models use facial landmarks to detect the driver's eye state, while the YOLOv8 model detects the head posture and presence of other objects that could obstruct the driver's view. All the models generate probability scores that indicate the level of drowsiness. The drowsiness possibility from all the models is then combined using Hard Voting to obtain the final drowsiness prediction.
- **Output:** The output component consists of audio alerts. If the drowsiness score exceeds a predefined threshold, an audio alert, such as an alarm or voice command, is triggered to prompt the driver to take a break or stop the vehicle.

3.2.2. Alcohol Detection Module:

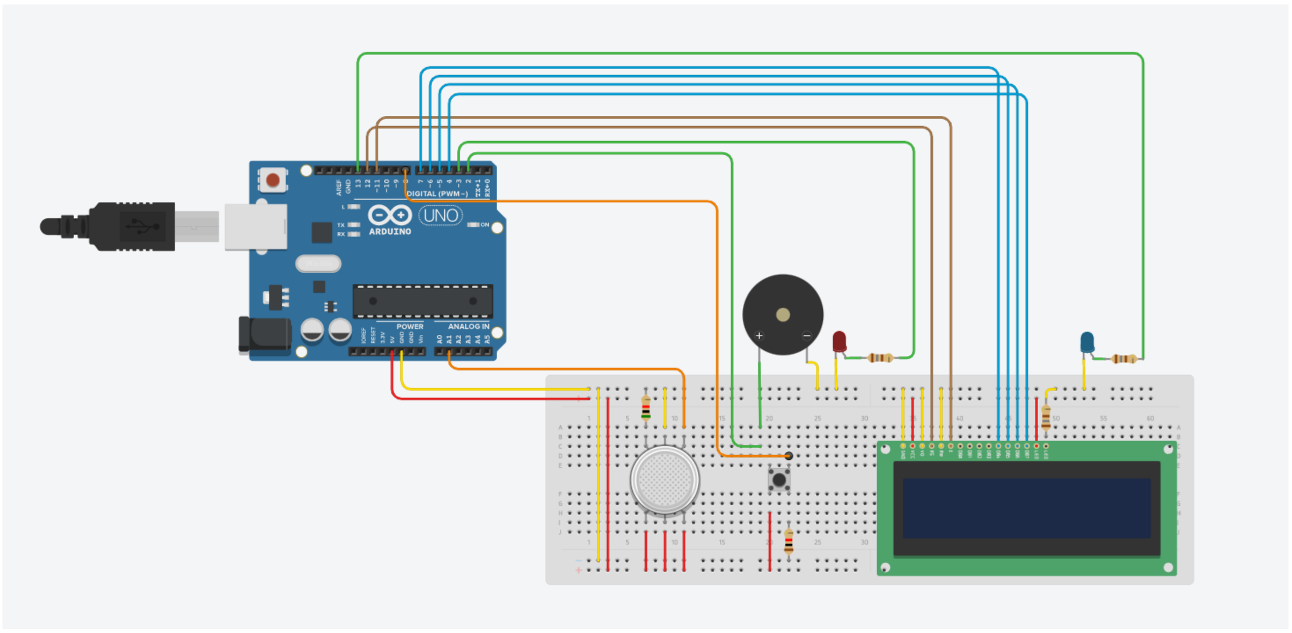


Fig-3: System Architecture of Alcohol Detection

In this project, we propose an architecture that uses an MQ-3 sensor connected to an Arduino UNO-3 to detect the level of alcohol in a person's breath. The system triggers a piezo buzzer and an LED if a high level of alcohol is detected. It also displays a message on an LCD screen, giving the driver an option to enable the emergency mode if necessary. The system architecture consists of five main components: input, pre-processing, decision-making, output, and emergency mode.

❖ **Input:** The input component comprises an MQ-3 sensor that measures the level of alcohol in a person's breath. The sensor is connected to an analog pin of an Arduino UNO-3.
❖ **Pre-processing:** The pre-processing component performs data cleaning to remove any noise from the analog signal. It then applies a calibration process to obtain an accurate reading of the alcohol level.
❖ **Decision Making:** The decision-making component compares the alcohol level against a predefined threshold. If the alcohol level is below the threshold, the system does nothing. If the alcohol level is above the threshold, the system triggers a piezo buzzer and an LED to alert the driver. It also displays a message on an LCD screen, giving the driver an option to enable the emergency mode if necessary.
❖ **Output:** The output component consists of three parts: piezo buzzer, LED, and LCD screen. The piezo buzzer and LED are triggered when the alcohol level is above the threshold. The LCD screen displays a message informing the driver about the alcohol level and giving them the option to enable emergency mode.
❖ **Emergency Mode:** The emergency mode component consists of a pushbutton that enables emergency mode if pressed. When the pushbutton is pressed, the LCD screen displays a message asking the driver to confirm whether they want to enable emergency mode or not. If the driver confirms, the system sends a signal to the car's central control system to stop the Alcohol Detection Process since it’s an Emergency Case.

3.3. Proposed System Model

The methodology adopted for DriveSafe360 is designed to efficiently integrate deep learning and IoT technologies to prevent accidents by detecting drowsiness and alcohol consumption in drivers and monitoring speed limits in real-time. The system comprises two main modules: Human Fatigue (Drowsiness) detection, and Alcohol Detection.

3.3.1. Human Fatigue Detection Module:

1. Load and Annotate Image Dataset:
    - Begin by loading the image dataset containing images of drivers into the Roboflow platform.
    - Annotate each image by marking whether the driver is drowsy or awake in the dataset. This annotation process helps in training the model to recognize drowsiness accurately.

2. Perform Training and Create the YOLOv8 Model:
   - Utilize the annotated images to train a YOLOv8 model, which is a deep learning-based object detection algorithm.
   - The training process involves feeding the annotated images to the model and optimizing its parameters to learn the patterns associated with drowsy and awake states.

3. Predict Drowsiness using Multiple Models:
   - During the prediction phase, the system determines whether the driver is drowsy or not based on the combined outputs of three different models.

   a. YOLO Model Prediction:
      - Utilize the pre-trained YOLOv8 model to predict whether the driver is drowsy.
      - The YOLO model examines the driver's facial features and detects signs of drowsiness based on learned patterns.

   b. Dlib Model Prediction:
      - Employ Dlib, a library for facial landmark detection, to extract the facial features of the eyes using facial landmarks.
      - Calculate the eye aspect ratio (EAR) by analyzing the eye's geometric properties.
      - If the EAR falls below a predefined threshold (30), predict that the driver is drowsy; otherwise, consider them awake.

   c. Cvzone Facemesh Model Prediction:
      - Use Cvzone, a computer vision library, to extract the facial features of the eyes and mouth using facial landmarks.
      - Calculate the eye aspect ratio (EAR) and mouth aspect ratio (MAR) for eyelid drooping & yawning detection respectively.
      - If the EAR is below a specified threshold (50), predict that the driver is drowsy; otherwise, consider them awake.
      - If the MAR is above a specified threshold (60 => Yawning MAR), predict that the driver is drowsy; otherwise, consider them awake.

4. Ensembling (Majority Voting) the Model Results:
   - Combine the predictions from the three models (YOLO, Dlib, and Cvzone) using a majority voting approach.
   - If the majority of models predict drowsiness, output that the driver is drowsy; otherwise, consider them awake.

5. Continuous Drowsiness Prediction:
   - To enhance accuracy and avoid false alarms, track the predictions over a time window.
   - If the driver is predicted to be drowsy for a continuous sequence of 30 frames, classify them as drowsy.
   - This approach ensures that momentary blinks or facial movements do not lead to false drowsiness predictions.

By following this step-by-step procedure, the system can effectively load and annotate the image dataset, train the YOLOv8 model, and utilize multiple models to predict driver drowsiness. The ensembling of the model results, combined with continuous drowsiness prediction, enables accurate and reliable identification of drowsy drivers.

3.3.2. Alcohol Detection Module:

1. Establish Circuit Connections:
   - Connect the Arduino UNO to the circuit by ensuring all the necessary connections are made correctly.
   - Connect the LCD screen, LEDs, Buzzer, Push Button, and MQ3 alcohol sensor to the Arduino UNO using jumper wires.

2. Connect Power Supply:
   - Provide power to the circuit by connecting the appropriate power supply.
   - Ensure the power supply is stable and capable of powering all the components.

3. Continuous Alcohol Monitoring:
   - Set up the circuit to continuously monitor the presence of alcohol using the MQ3 alcohol sensor.
   - The sensor should be configured to detect any alcohol vapors in the environment.

4. Trigger Red LED and Piezo Buzzer:
   - If the MQ3 sensor detects alcohol presence, it immediately triggers the Red LED and the Buzzer to indicate the presence of alcohol.
   - The Red LED and Buzzer should remain active until there is no presence of Alcohol detected.

5. Display Message on LCD Screen:
    - After the 20-second alarm, display a message on the LCD screen asking whether to activate the emergency mode.
    - The LCD screen should prompt the user with an appropriate message and await their response.

6. Countdown Timer:
    - Start a countdown timer on the LCD screen, initially set to 20 seconds.
    - The timer should decrease by one second at a time until it reaches 0 secs.

7. Emergency Mode Activation:
    - If the alcohol-impaired driver finds themselves in an emergency, they can activate the emergency mode.
    - To activate the emergency mode, the driver should press the push button within the 20-second timeframe.
    - If the push button is pressed, the emergency mode will be activated, Deactivating the Red LED and Buzzer, and activating another Blue LED (Indicating Emergency Mode).

8. Engine Lock:
    - If the push button is not pressed within the 20-second timeframe, then display a message on the LCD screen stating that the engine is locked.
    - The engine lock message serves as a safety measure to prevent the impaired driver from driving the vehicle.
    - If the User of the vehicle wants to avoid this engine lock situation, then there should be no alcohol presence near the dashboard (where MQ-3 Sensor will be placed), which means that the alcohol-impaired person should move out from the driver seat as soon as possible before this Engine Lock situation takes place.

By following this step-by-step procedure, you can establish the necessary circuit connections, monitor alcohol presence, trigger appropriate alerts, display messages on the LCD screen, activate the emergency mode, and implement an engine lock if required. This system helps ensure safety by alerting and preventing alcohol-impaired individuals from driving and provides an option for emergency situations.

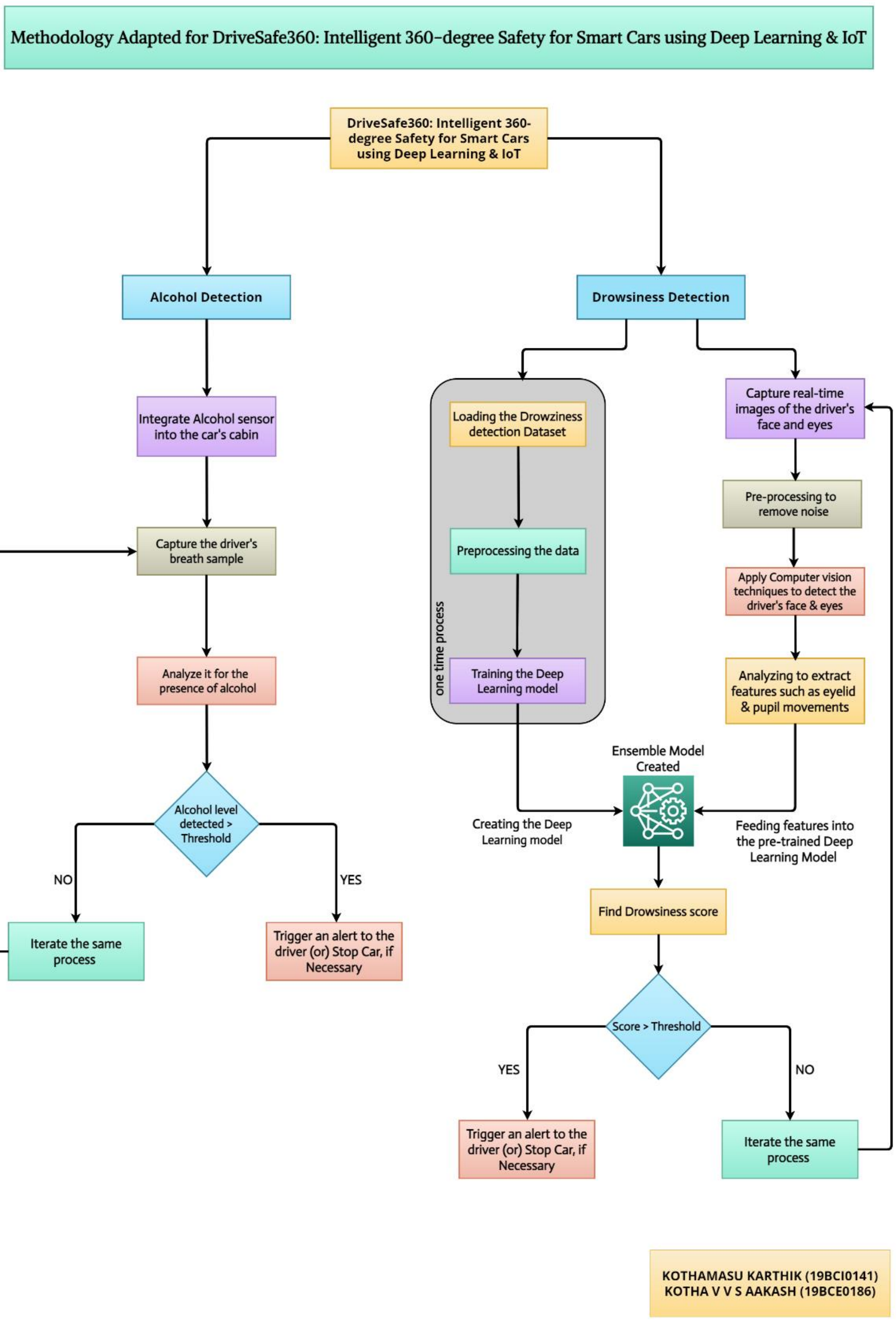

Methodology Adapted for DriveSafe360: Intelligent 360-degree Safety for Smart Cars using Deep Learning & IoT
DriveSafe360: Intelligent 360-degree Safety for Smart Cars using Deep Learning & IoT
Alcohol Detection
Drowsiness Detection
Integrate Alcohol sensor into the car's cabin
Capture the driver's breath sample
Analyze it for the presence of alcohol
Alcohol level detected > Threshold
NO
YES
Iterate the same process
Trigger an alert to the driver (or) Stop Car, if Necessary
one time process
Loading the Drowziness detection Dataset
Preprocessing the data
Training the Deep Learning model
Capture real-time images of the driver's face and eyes
Pre-processing to remove noise
Apply Computer vision techniques to detect the driver's face & eyes
Analyzing to extract features such as eyelid & pupil movements
Ensemble Model Created
Creating the Deep Learning model
Feeding features into the pre-trained Deep Learning Model
Find Drowsiness score
Score > Threshold
YES
NO
Trigger an alert to the driver (or) Stop Car, if Necessary
Iterate the same process
KOTHAMASU KARTHIK (19BCI0141)
KOTHA V V S AAKASH (19BCE0186)

Fig-4: Proposed System Model for DriveSafe360

4. PROPOSED SYSTEM ANALYSIS AND DESIGN

4.1. Introduction:

The proposed system, "DriveSafe360: Intelligent 360-degree Safety for Smart Cars using Deep Learning & IoT," aims to enhance road safety by integrating advanced technologies and intelligent algorithms. The project includes two modules: Human Fatigue (Drowsiness) detection and Alcohol Detection. The system's objective is to develop a real-time solution that can monitor driver fatigue and detect alcohol consumption. By leveraging deep learning models like Dlib, YOLOv8, and cvzone, the system can accurately identify signs of drowsiness, issuing timely warnings to the driver.

The integration of IoT enables seamless detection of alcohol-impaired driving. The project's scope covers algorithm development, IoT implementation, and integration into a user-friendly system. The system combines facial landmark detection, eye state recognition, and alcohol vapor measurement using the MQ-3 sensor. The interface will display alerts on the car's dashboard and provide comprehensive reports for long-term monitoring. The proposed system aims to improve road safety and prevent accidents by leveraging intelligent technologies for smart car safety.

4.2. Requirement Analysis:

The requirements analysis section breaks down the various requirements for the proposed system, including its functional features and non-functional characteristics such as usability and sustainability.

4.2.1. Functional Requirements:

4.2.1.1. Product Perspective:

Overall, DriveSafe360 is designed to provide a comprehensive safety solution that combines advanced technologies with ease of use. By addressing key safety factors such as driver fatigue and alcohol impairment, the product aims to enhance driver safety and prevent accidents, ultimately contributing to a safer and more secure driving experience.

To create the Driver fatigue Detection module, we have used the DLIB, cvzone, and YOLO algorithm and for Alcohol Detection Module we have used the Arduino uno r3 and Mq3 alcohol sensor.

4.2.1.2. Product features:

- ❖ **Driver fatigue detection:** This component consists of audio alerts. If the driver fatigue score exceeds a predefined threshold, an audio alert, such as an alarm or voice command, is triggered to prompt the driver to take a break or stop the vehicle.
- ❖ **Alcohol Detection:** The system triggers a piezo buzzer and an LED if a high level of alcohol is detected. It also displays a message on an LCD screen, giving the driver an option to enable the emergency mode if necessary.
- ❖ **Ensemble Model:** The ensemble model combines the outputs of two deep learning models, Dlib, Cvzone and YOLOv8, to improve the detection accuracy. The Dlib model uses facial landmarks to detect the driver's eye state, while the YOLOv8 model detects the head posture and presence of other objects that could obstruct the driver's view. Both models generate probability scores that indicate the level of drowsiness. The probability scores from both models are then combined using weighted averaging to obtain the final drowsiness score. The weightage for each model is adjusted based on their performance on the validation dataset.
- ❖ **Alert message:** When drowsiness is detected, the system can provide the driver visual and audio indications, such as a warning buzzer (Beep sound) , or a message on the dashboard.
- ❖ **Customizable dataset training:** It gives greater flexibility in the model training procedure. Compared to what would be possible with a more general dataset, it is likely to get more accuracy and better performance.
- ❖ **Frame to Frame detection:** It allows for the automatic detection of objects within a video sequence and can be additionally utilized for alert operators.

4.2.1.3. User characteristics:

- ❖ **Movement of the eye:** When a driver is drowsy, their eye movements tend to slow down, and they may close their eyes for longer periods of time. A driver fatigue detection system may use infrared cameras to monitor these eye movements.
- ❖ **Movement of head:** When a driver is sleepy, they may nod their head or tilt it up or down, then our system will be able to detect whether the driver is drowsy if the head tilts down or up.

4.2.1.4. Assumption & Dependencies:

- ❖ The camera utilized for capturing the driver's face and the sensor used for detecting alcohol is of enough resolution to gather the information required for driver fatigue and alcohol detection.

- ❖ Throughout the driving session, the driver's face is visible to the camera.
- ❖ The YOLOv8 model's training data accurately represents the variety of facial expressions and other elements that may impact drowsiness detection.
- ❖ Our System depends upon lighting inside the vehicle.

4.2.1.5. Domain Requirements:

- ❖ System should be connected to the internet.
- ❖ **Data Collection:** A large collection of photos is required to train the YOLOv8 model for fatigue detection. The dataset should include a variety of lighting conditions, face expressions, and other variables that could impact driver fatigue detection.
- ❖ Since the YOLOv8 technique is computationally complex, it requires a powerful GPU to execute efficiently.

4.2.1.6 User Requirements:

- ❖ **Alert message:** Our system will be able to detect whether the driver is drowsy or whether the driver consumes alcohol.
- ❖ **Clear indicators:** The system should give the driver obvious indicators when something is wrong, such as alert warnings.
- ❖ **Reliable:** The system shouldn't generate false results, as they could result in pointless alerts or missed detections.

4.2.2. Non-Functional Requirements:

The non-functional requirements section breaks down the various requirements for the proposed system, including its product requirements and organizational requirements such as deployment requirements and engineering standard requirements.

4.2.2.1 Product Requirements:

4.2.2.1.1 Efficiency (in terms of Time and Space)

Our system uses the YOLOv8 algorithm, when it comes to other techniques, our model gives results very fast and accurately. When it comes to speed, it can process over 40 frames per second on a GPU, making it perfect for high-speed applications. It is meant to be easy to train, which gives it an important edge over other object identification algorithms. It just requires one training session and may be learned on a minimal dataset.

4.2.2.1.2 Reliability:

Our model is very reliable on the dataset that we are creating. Since, we created our own dataset so that it will be able to detect easily.

4.2.2.1.3 Portability:

The proposed system has been designed in such a way that it can be used in different types of vehicles. The device is compact having a dimension of 13.5cm x 27cm x 27cm making it easy to carry and transport. The system's inbuilt light ensures that it can be used in different lighting conditions & making it a versatile tool for drivers.

4.2.2.1.4 Usability:

To apply this model, we need a camera and alcohol detect sensor in a real- world scenario. It needs to be placed somewhere near to the driver and also have less obstacles to the driver like inside steering, or in the dashboard which is right opposite to the driver. Also, the device requires minimal maintenance which further reduces the overall cost of ownership.

4.2.2.2 Organizational Requirements:

4.2.2.2.1 Implementation Requirements (in terms of deployment):

- **System Integration:** To apply this model, we need a camera and alcohol detect sensor in a real- world scenario. It needs to be placed somewhere near to the driver and also have less obstacles to the driver like inside steering, or in the dashboard which is right opposite to the driver.
- **Testing and Validation**: To make sure the system satisfies the operational and safety criteria; it should go through thorough testing and validation.
- **Training:** Instruction on how to use and maintain the system should be provided to drivers and maintenance staff.
- **Maintenance and Support:** To guarantee the system's continuing operation and dependability, it should be maintained and supported.
- **Performance Monitoring**: To make sure the system is fulfilling its operating needs and to find areas for improvement, performance should be tracked.

4.2.2.2.2 Engineering Standard Requirements:

- The device should be designed & manufactured to comply with relevant engineering standards such as electrical safety standards and compatibility standards.
- The device should be tested to ensure that it meets the required performance specifications such as accuracy & speed of image processing and laterality detection.
- The device should be designed to minimize the risk of failure during usage with appropriate safety features incorporated.
- The device should be designed to be easy to use and understand with a user-friendly interface and clear instructions for use provided.

4.2.2.3. Operational Requirements:

- **Economic:** The proposed system is designed to provide a cost-effective solution to reduce accidents. The system uses a piezo buzzer and an LED which are affordable & readily available components.
- **Environmental:** The device does not have any significant environmental impact. It doesn’t require any hazardous material to operate & its low power consumption helps to reduce environmental impact.
- **Social:** The proposed system provides a valuable addition to the motor field industry by offering a user-friendly interface & easy integration. The system should help society by lowering the number of accidents brought on by drunk driving and raising traffic safety.
- **Political:** The proposed system is not subject to any political regulations or requirements.
- **Ethical:** The proposed system is designed to ensure that to reduce the no of accidents which is happened due to driver fatigue and also drunk driving. The system complies with ethical principles & guidelines in the motor field industry.
- **Health and Safety:** There won’t be any danger to the driver or passengers from our system. It won’t interfere with the driver's capacity to control the car safely. The health of the driver won’t be harmed by our system.
- **Sustainability:** It uses low-power components reducing energy usage. The system also offers an effective way to reduce accidents during the journey which can lead to further complications, reducing the impact on the environment.
- **Legality:** The proposed system complies with legal regulations in the motor field industry to ensure driver data privacy and confidentiality.
- **Inspectability:** Proposed system is easily inspectable as it has a user-friendly interface & validation result can be sent to the LCD which can be easily reviewed and audited.

4.2.3 System Requirements:

The device works together with hardware & software components to achieve this goal.

4.2.3.1. S/W Requirements:

- PyCharm
- Python Libraries: OpenCV, Dlib, Yolov8, face_utils, cvzone.FaceMeshModule, ultralytics (YOLO)
- Google Colab (For Training YOLOv8 Model)
- Roboflow (For Annotating the Image Dataset)
- Arduino IDE
- Tinkercad

4.2.3.2. H/W Requirements:

- Computer or Laptop which consists of working Operation System. (We used the Windows OS)
- Camera with night vision is highly preferable. (For Drowsiness Detection)
- Arduino Uno R3: The Arduino Uno R3 is being used as the main microcontroller board in this project. It provides a platform for controlling and interfacing with various components of the system. Arduino Uno is chosen for its simplicity, versatility, and ease of use. It enables the execution of the necessary code and facilitates communication between different parts of the circuit.

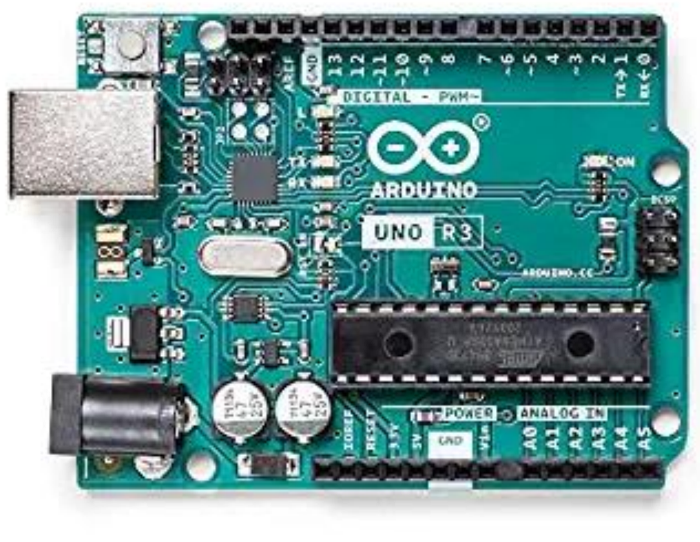


Fig-5: Arduino Uno R3

❖ MQ-3 alcohol sensor: The MQ-3 alcohol sensor is a key component of the project as it is responsible for detecting the presence of alcohol vapors in the environment. This sensor uses a semiconductor to measure the concentration of alcohol in the surrounding air. It plays a critical role in determining whether the driver is under the influence of alcohol, allowing for timely alerts and appropriate actions to be taken.

Fig-6: MQ-3 alcohol sensor

❖ Piezo buzzer: The piezo buzzer is used to generate audible alerts or alarms in the system. In this project, the buzzer is triggered when alcohol vapors are detected by the MQ-3 sensor. It provides an audible indication to both the driver and those nearby that alcohol is present, signaling the need for caution or action.

Fig-7: Piezo buzzer

❖ LED’s: LEDs (Light Emitting Diodes) are employed as visual indicators in the project. They serve as warning lights to visually alert individuals about the presence of alcohol. The LEDs are typically activated in conjunction with the piezo buzzer to provide a combined visual and audible warning.

Fig-8: LED

- Lcd Screen 16x2: The LCD screen (Liquid Crystal Display) is used to provide visual feedback and display relevant information to the user. It can show messages, prompts, and countdown timers. In this project, the LCD screen is utilized to display messages regarding the presence of alcohol, emergency mode activation, and engine lock status.

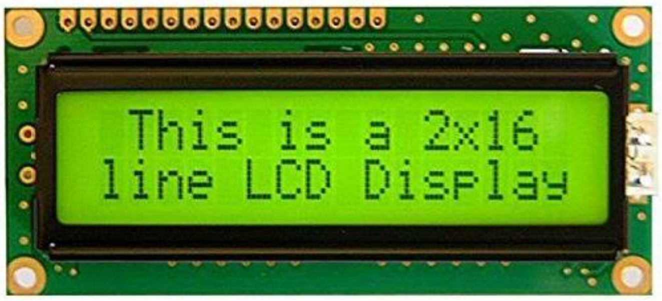


Fig-9: LCD Screen 16 x 2

- Push button: The push button is a user input component used for manual activation of the emergency mode. It allows the driver, in case of an emergency situation, to indicate the need for immediate assistance or special action. By pressing the push button within a specified timeframe, the driver can activate the emergency mode.

Fig-10: Push button

- Few 180 Ω Resistors: The 180 Ω resistors are used as current-limiting resistors in conjunction with the LEDs. They protect the LEDs from excessive current flow, preventing damage to the LEDs and ensuring proper operation. The resistors help regulate the flow of electricity, allowing the LEDs to emit light at the desired brightness level.

Fig-11: 180 Ω Resistors

5. RESULTS AND DISCUSSIONS

5.1. Alcohol Detection:

Software Implementation:

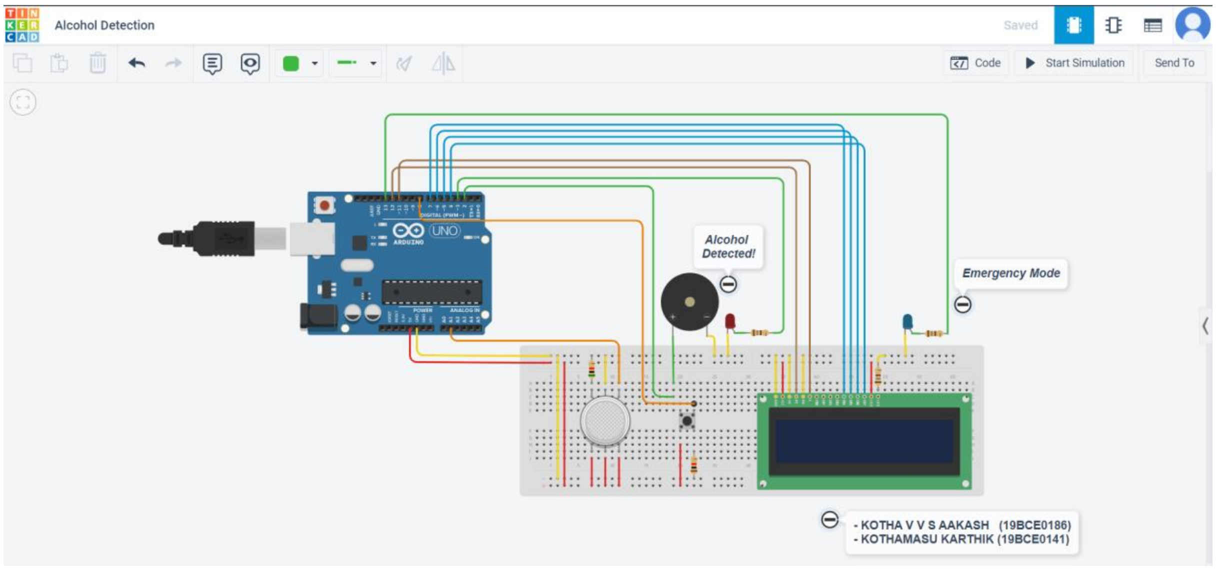


Fig: Entire Circuit Before Power Supply

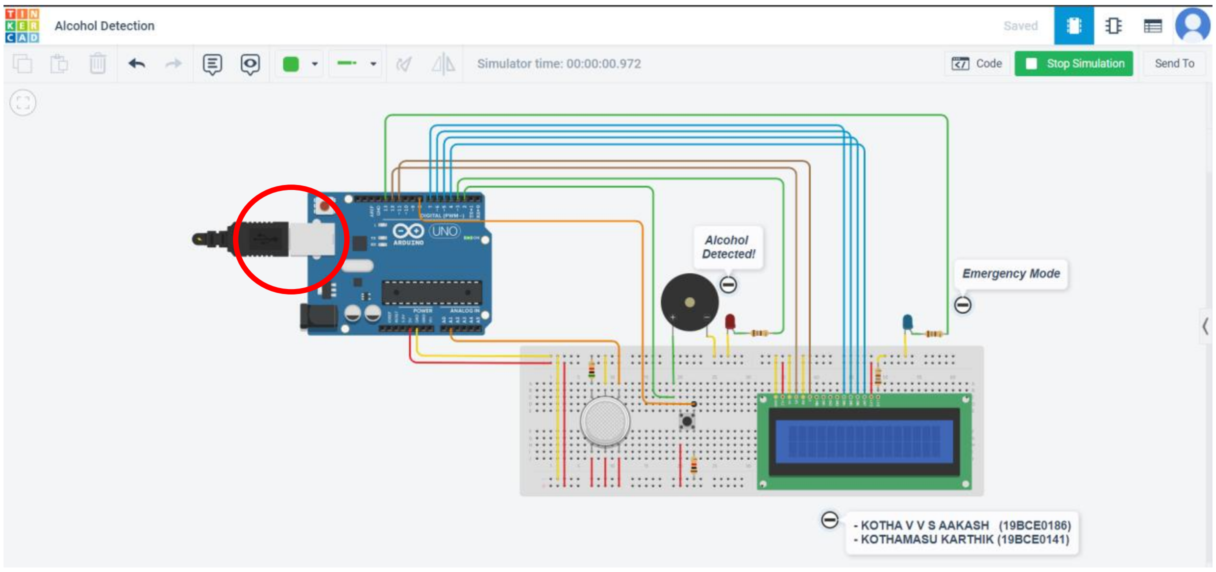

Alcohol Detection
Saved
Simulator time: 00:00:00.972
Code
Stop Simulation
Send To
Alcohol Detected!
Emergency Mode
- KOTHA V V S AAKASH (19BCE0186)
- KOTHAMASU KARTHIK (19BCE0141)

Fig: Entire Circuit After Power Supply

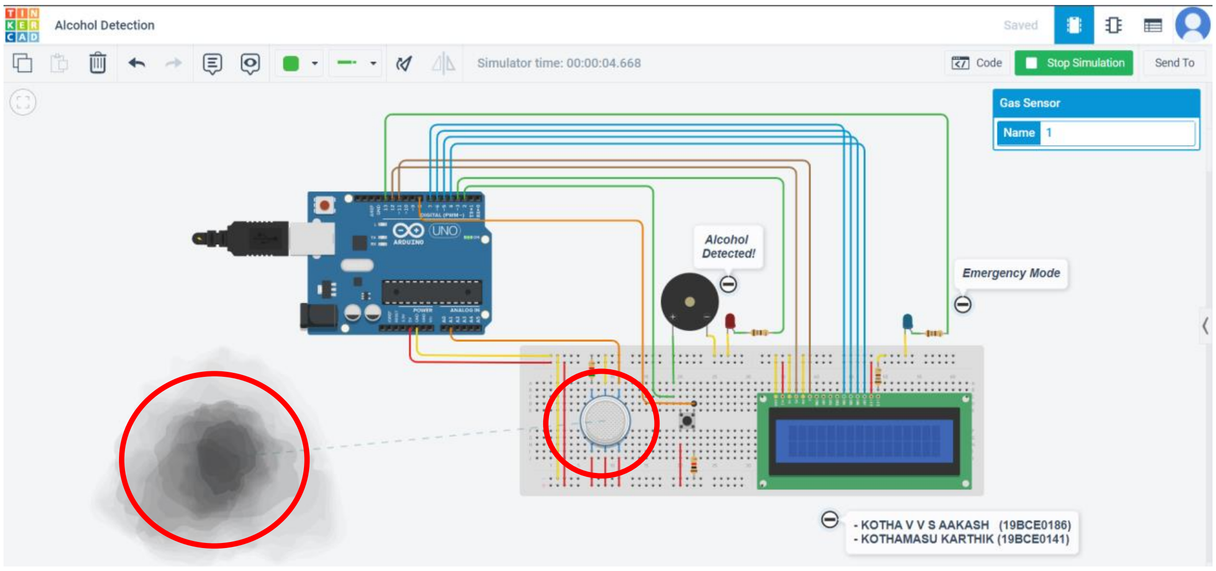


Fig: Alcohol is away from Gas Sensor

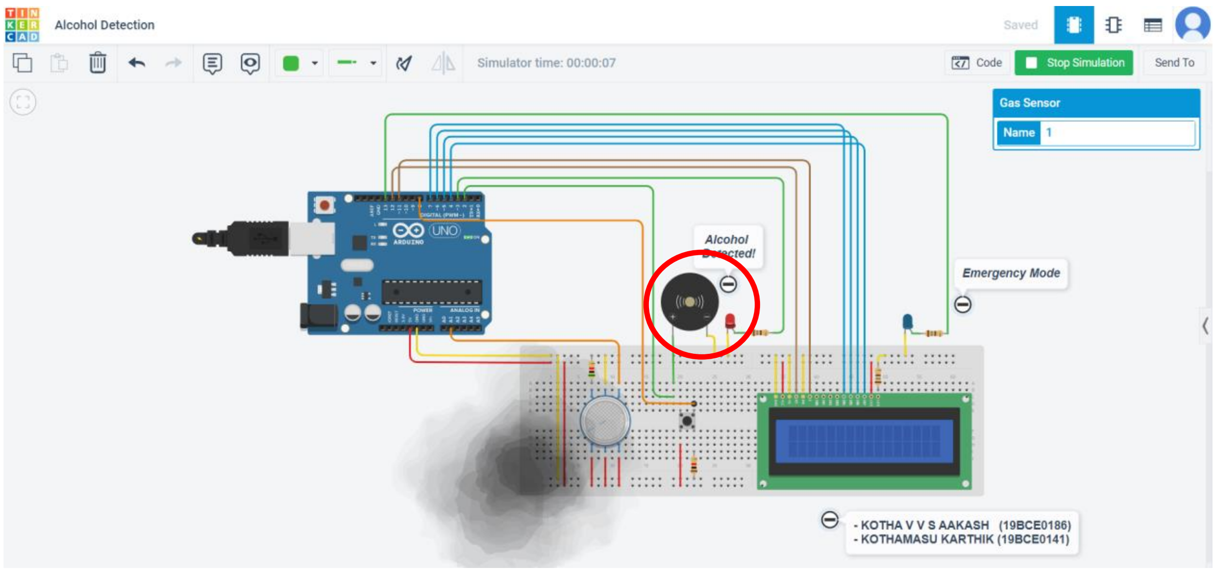


Fig: When Alcohol is near to Gas Sensor, Red LED & Piezo Buzzer are Triggered, indicating Danger and Alerting Driver

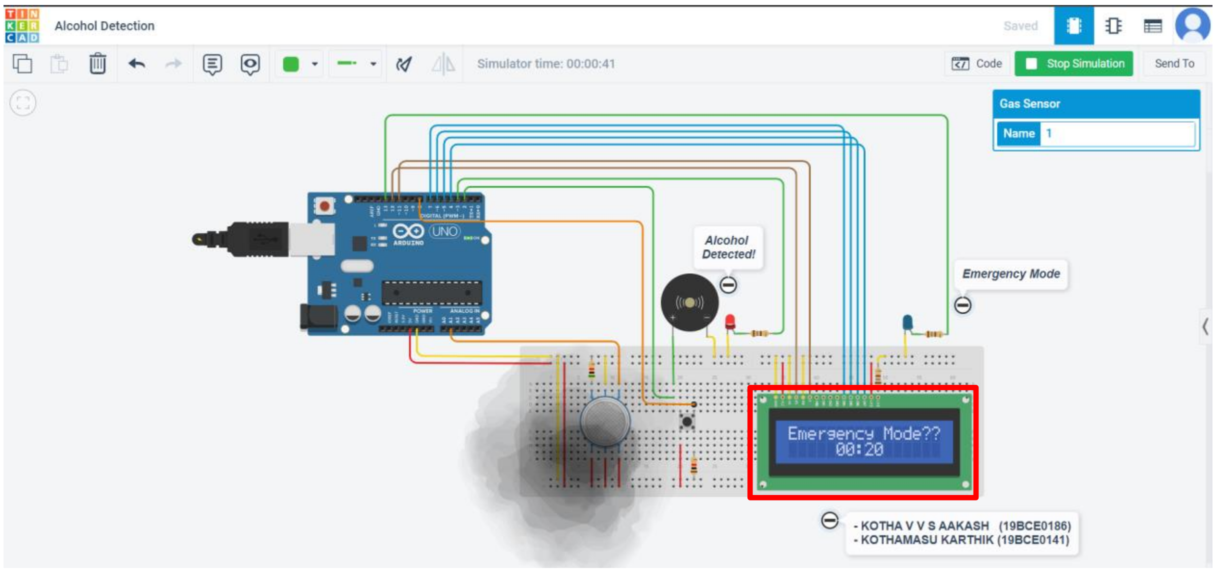


Fig: If Alcohol is near to Gas Sensor continuously for 20 Secs, LCD Screen will prompt the Driver if it’s required to enable Emergency Mode? (20 Secs Timer)

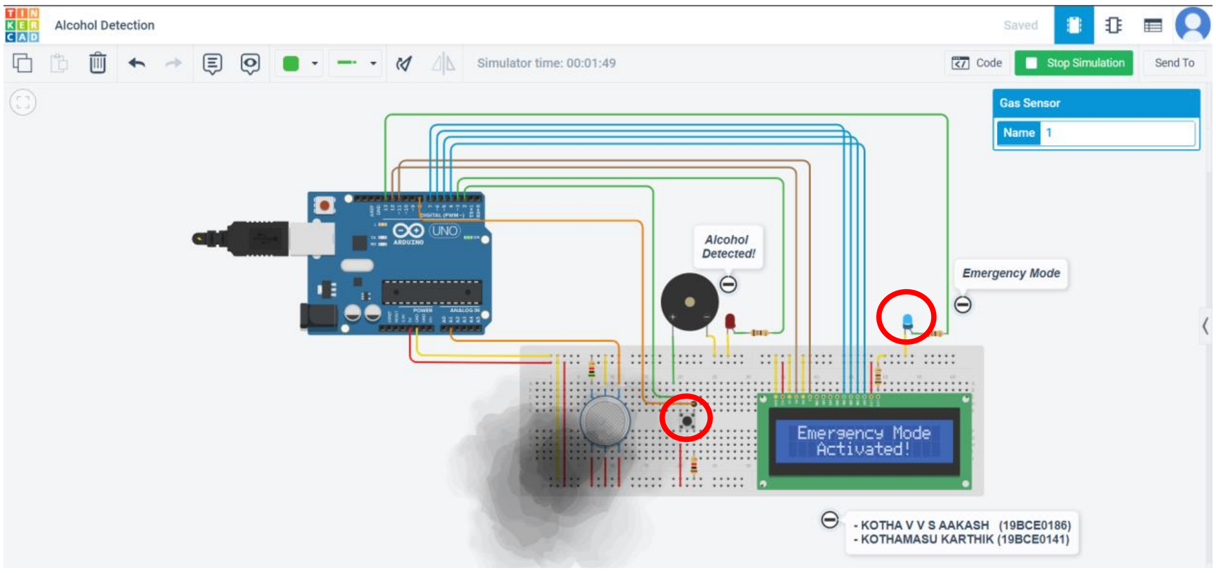


Fig: If the Alcohol impaired Driver is really in an emergency, he can turn on the Emergency mode in that 20 Secs Timer by pushing the Push Button, which trigger Blue LED indicating that Emergency Mode is Enabled

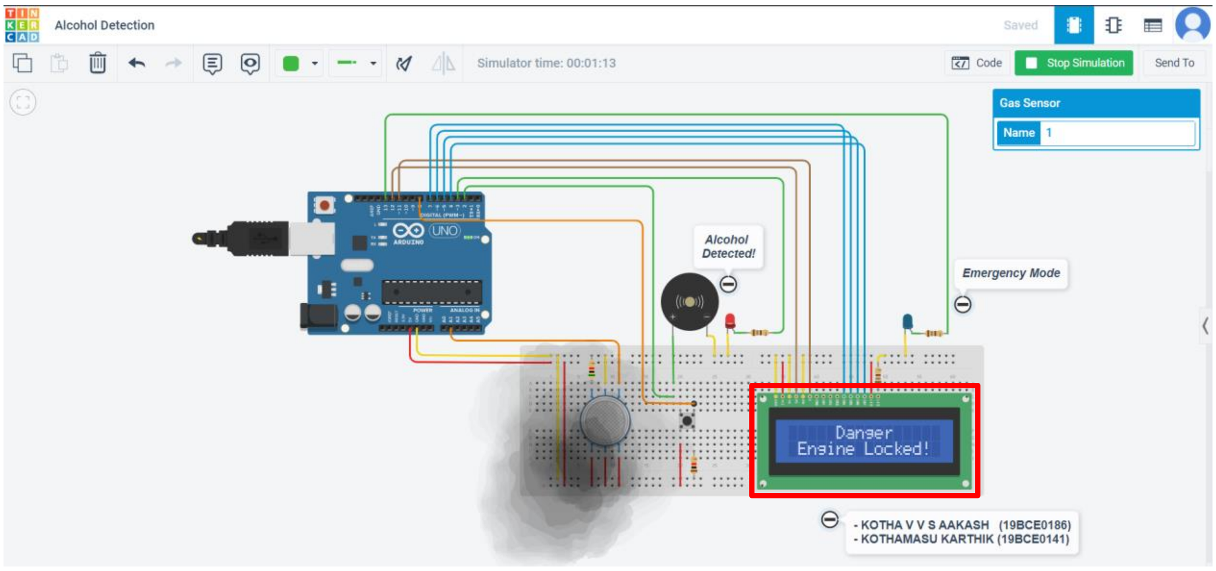


Fig: If the Emergency Mode is not Enabled in that 20 Secs Timer, Then after the Timer, Engine would be Locked for the safety of everyone

Hardware Implementation: (Demo: https://photos.app.goo.gl/wf9q5ZWBSF7RNEdT7)

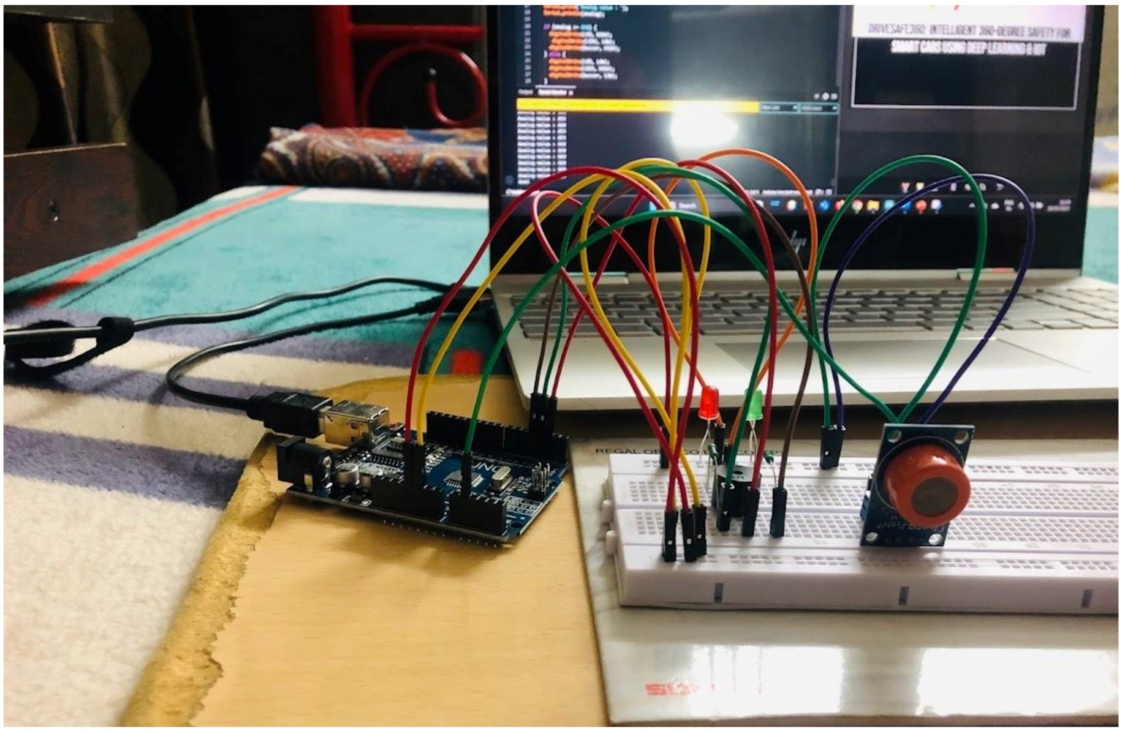

DRIVESAFE360: INTELLIGENT 360-DEGREE SAFETY FOR SMART CARS USING DEEP LEARNING & IOT

Fig: Entire Circuit Before Power Supply

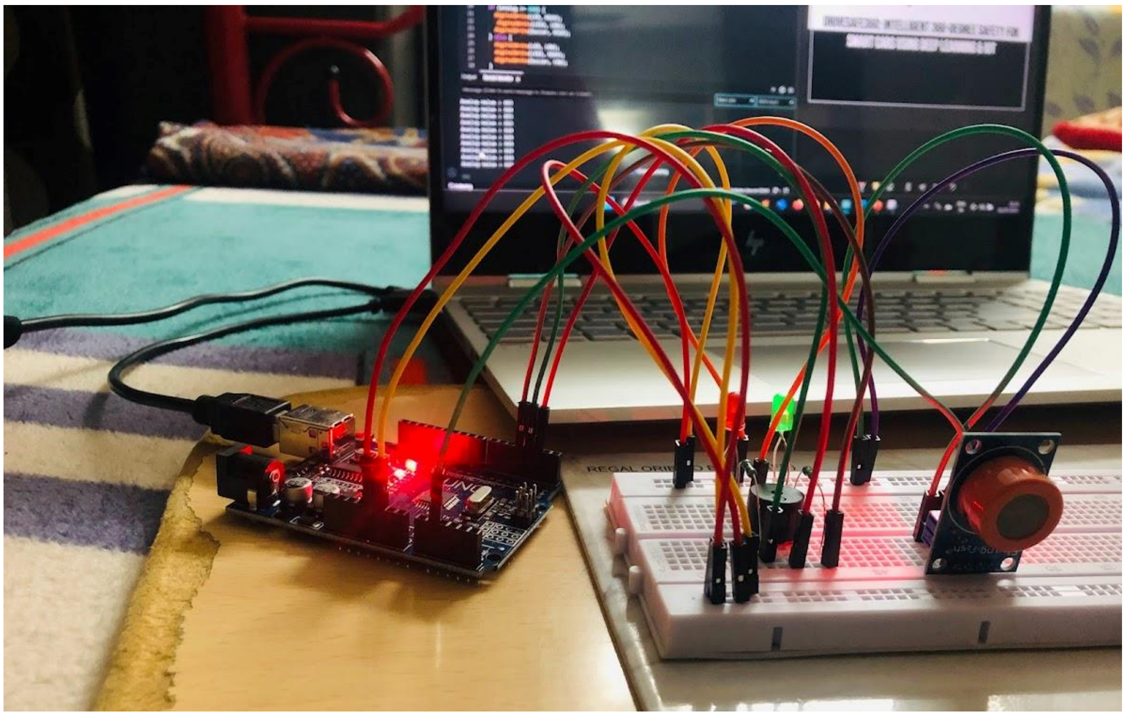

DRIVESAFE360: INTELLIGENT 360-DEGREE SAFETY FOR SMART CARS USING DEEP LEARNING & IOT

Fig: Entire Circuit After Power Supply, Green LED is turned on as there is no presence of alcohol near MQ-3 Sensor

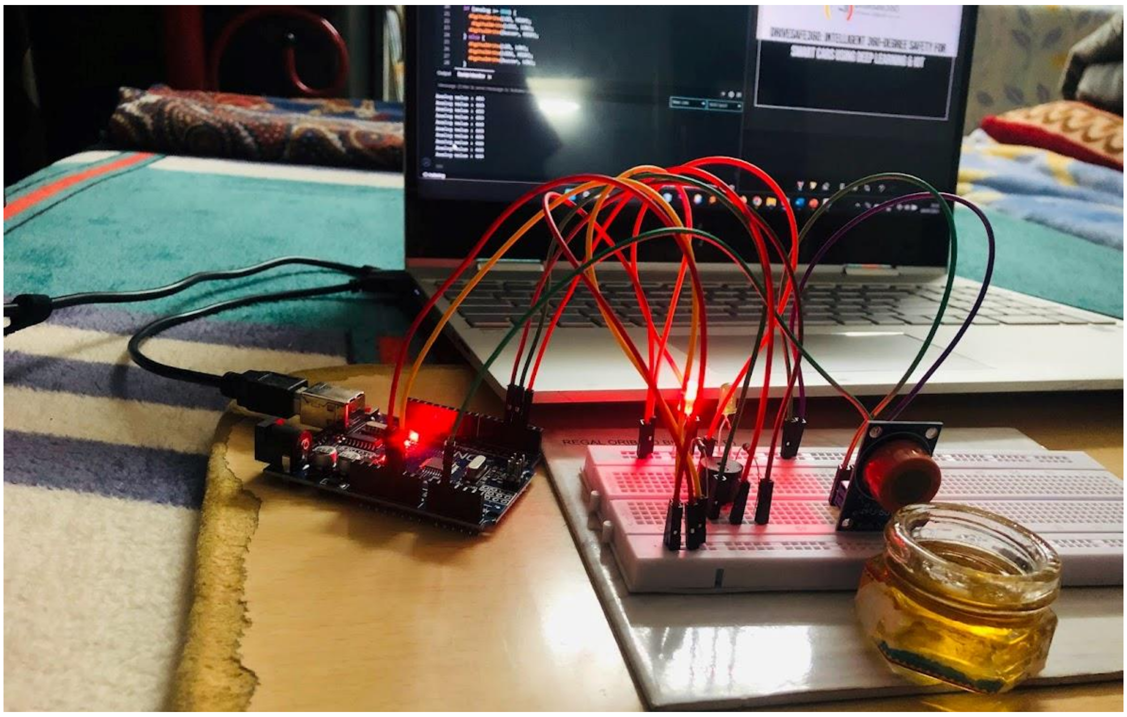


Fig: After Alcohol is brought closer to the MQ-3 Sensor, Red LED is turned on indicating Danger & Alerting the Driver

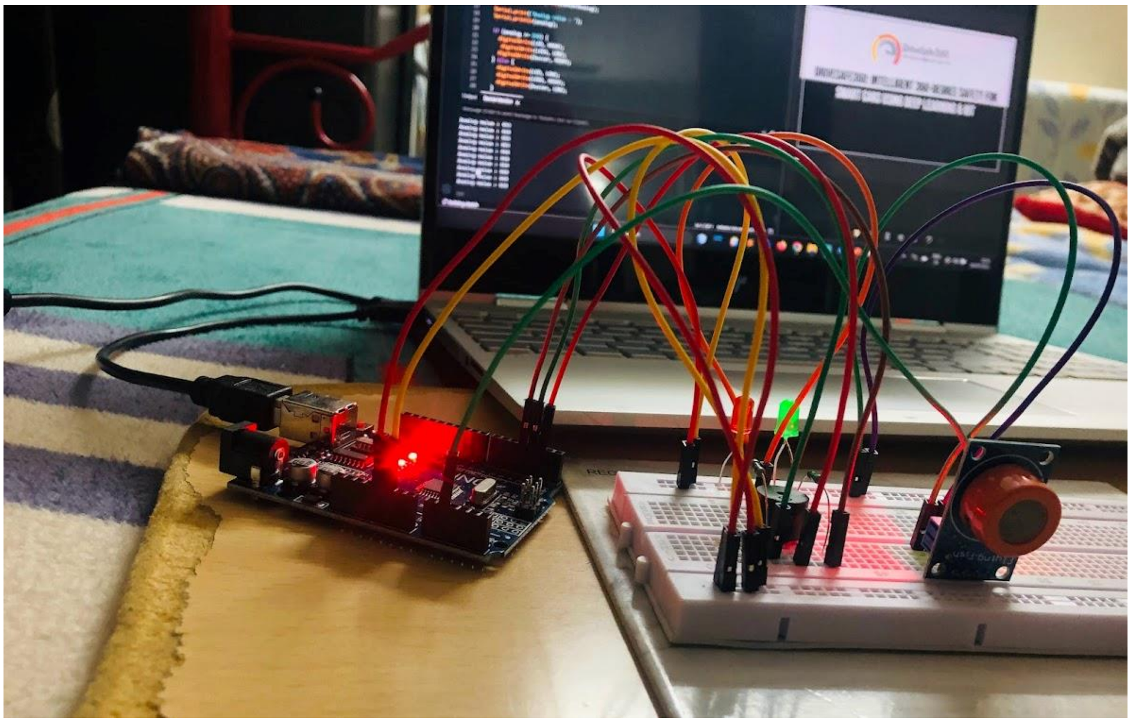
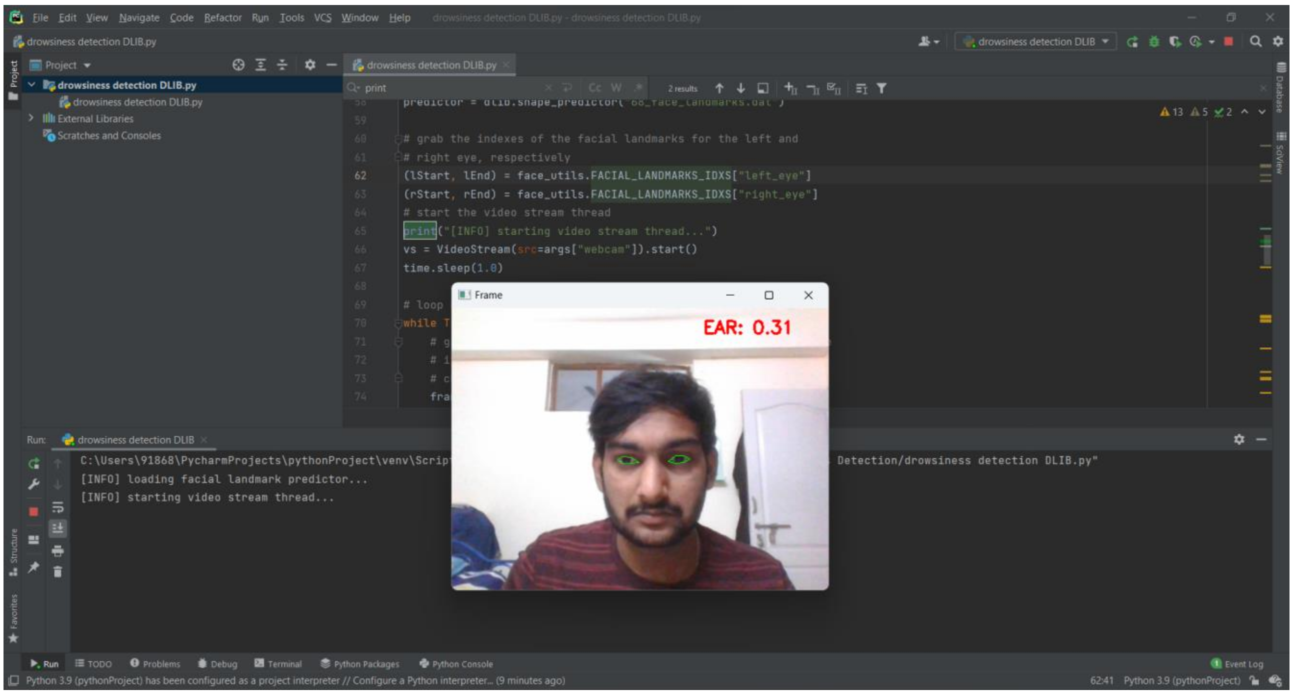


Fig: Once the Alcohol is taken away from the MQ-3 Sensor, Again Green LED is turned on indicating It is safer for Everyone.

Using this Hardware Prototype, we performed testing by moving Alcohol near and away from the MQ-3 Sensor for about 150 Times. In 146 of the times, the prototype gave the results as expected, and only about 4 times, the prototype performed with some huge delay. So, as the prototype we built worked 146/150 times correctly, the accuracy of this System will result as “**97.33%”.**

5.2. Human Fatigue Detection:

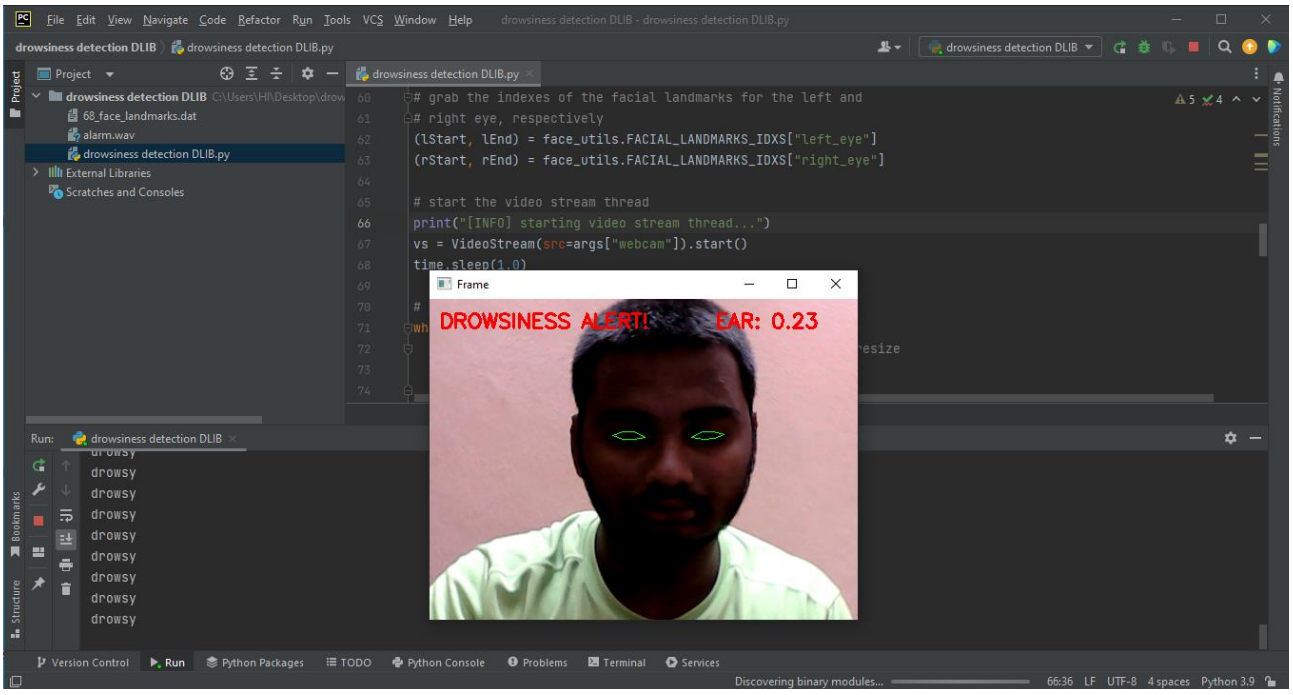
Human Fatigue Detection using Dlib:



```
predictor = dlib.shape_predictor("68_face_landmarks.dat")

# 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=args["webcam"]).start()
time.sleep(1.0)

# loop
while T
    # g
    # l
    # c
    fra
```

```
C:\Users\91868\PycharmProjects\pythonProject\venv\Scrip  Detection/drowsiness detection DLIB.py"
[INFO] loading facial landmark predictor...
[INFO] starting video stream thread...
```

Fig: Non-Drowsy/Awake since EAR > 30 (Threshold)

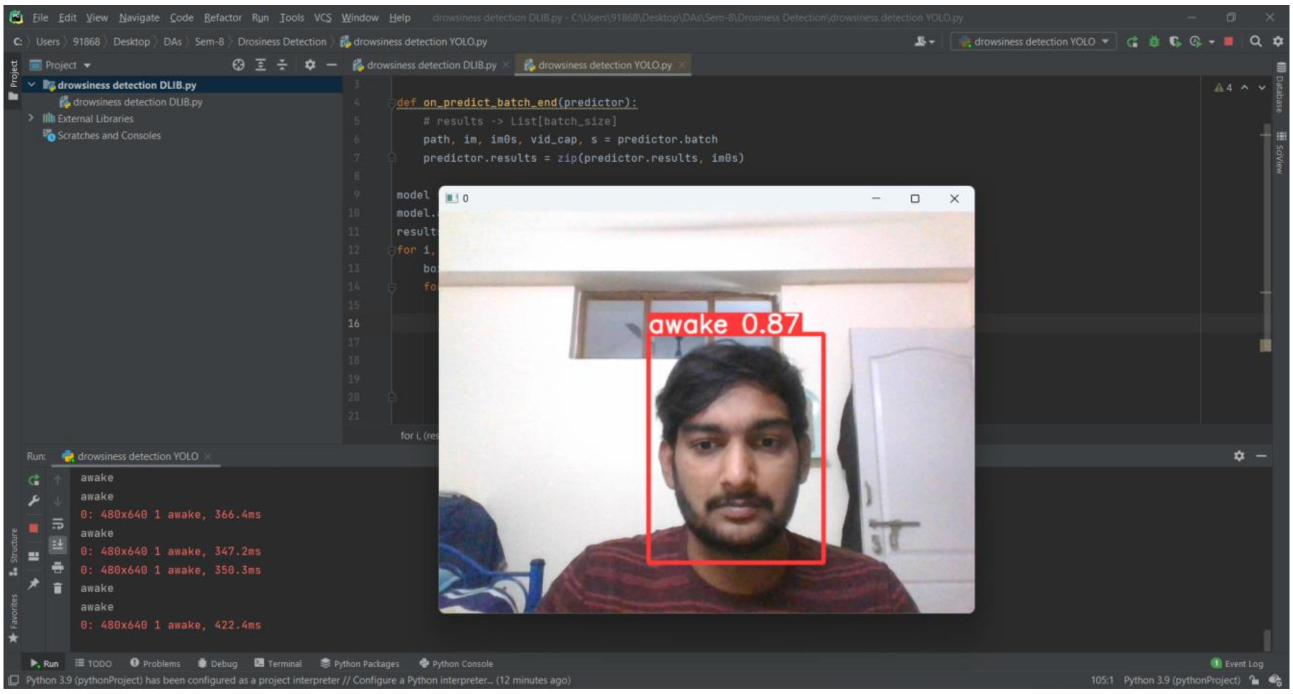


```
# grab the indexes of the facial landmarks for the left and
# right eye, respectively
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]

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

#
wh
                                                  resize
```

```
drowsy
drowsy
drowsy
drowsy
drowsy
drowsy
drowsy
drowsy
drowsy
```

Fig: Drowsy since EAR < 30 (Threshold)

Human Fatigue Detection using YOLOv8 Pretrained Model:

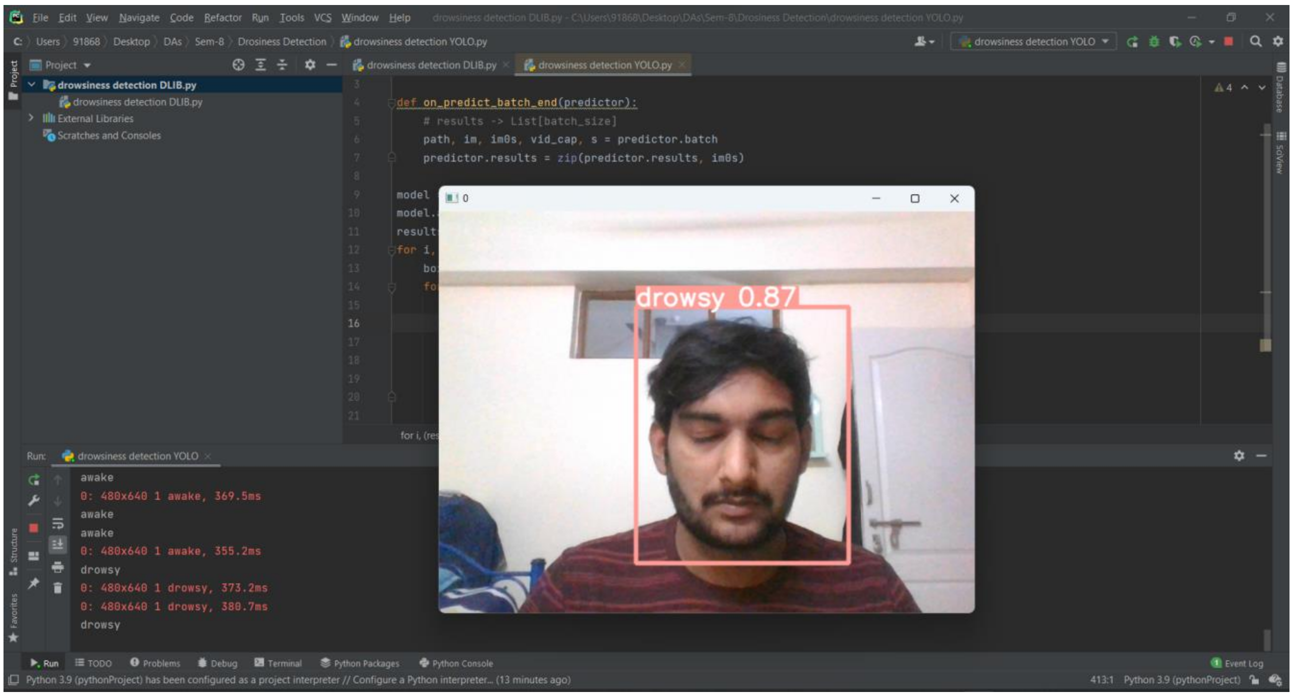


```
def on_predict_batch_end(predictor):
    # results -> List[batch_size]
    path, im, im0s, vid_cap, s = predictor.batch
    predictor.results = zip(predictor.results, im0s)

model
model.
result
for i,
    bo
    fo
```

```
awake
awake
0: 480x640 1 awake, 366.4ms
awake
0: 480x640 1 awake, 347.2ms
0: 480x640 1 awake, 350.3ms
awake
awake
0: 480x640 1 awake, 422.4ms
```

Python 3.9 (pythonProject) has been configured as a project interpreter // Configure a Python interpreter... (12 minutes ago)

Fig: Non-Drowsy/Awake



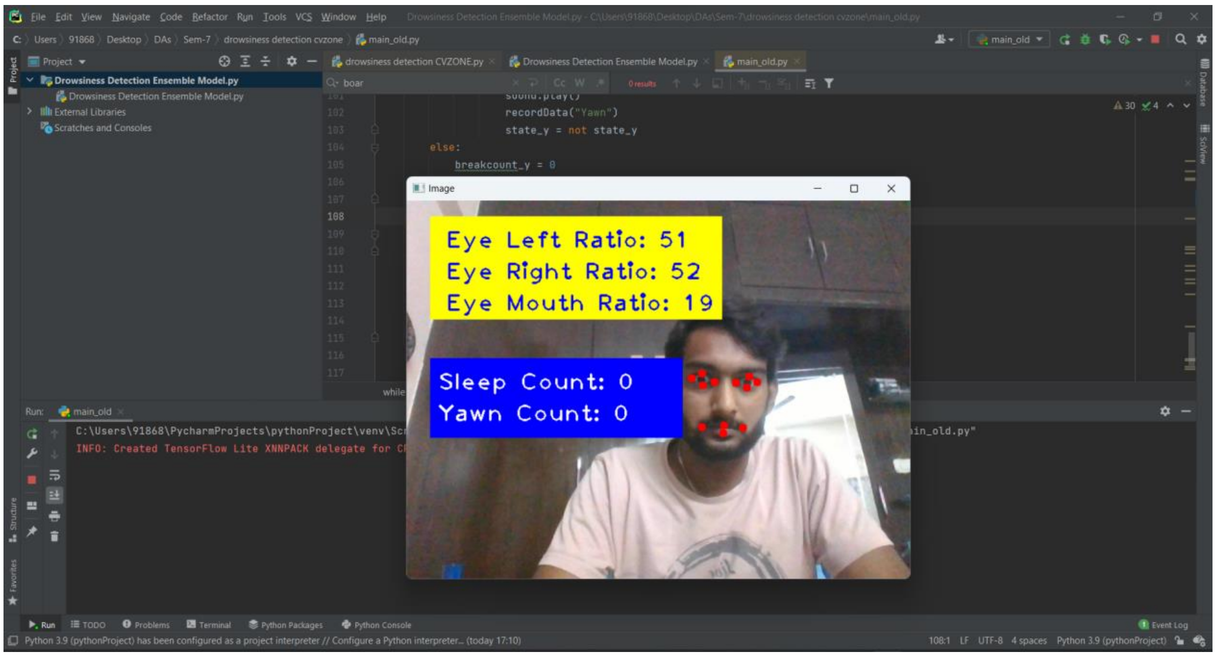
```
def on_predict_batch_end(predictor):
    # results -> List[batch_size]
    path, im, im0s, vid_cap, s = predictor.batch
    predictor.results = zip(predictor.results, im0s)

model
model.
result
for i,
    bo
    fo
```

```
awake
0: 480x640 1 awake, 369.5ms
awake
awake
0: 480x640 1 awake, 355.2ms
drowsy
0: 480x640 1 drowsy, 373.2ms
0: 480x640 1 drowsy, 380.7ms
drowsy
```

Python 3.9 (pythonProject) has been configured as a project interpreter // Configure a Python interpreter... (13 minutes ago)

Fig: Drowsy

Human Fatigue Detection using cvzone:



```
                sound.play()
                recordData("Yawn")
                state_y = not state_y
        else:
            breakcount_y = 0
```

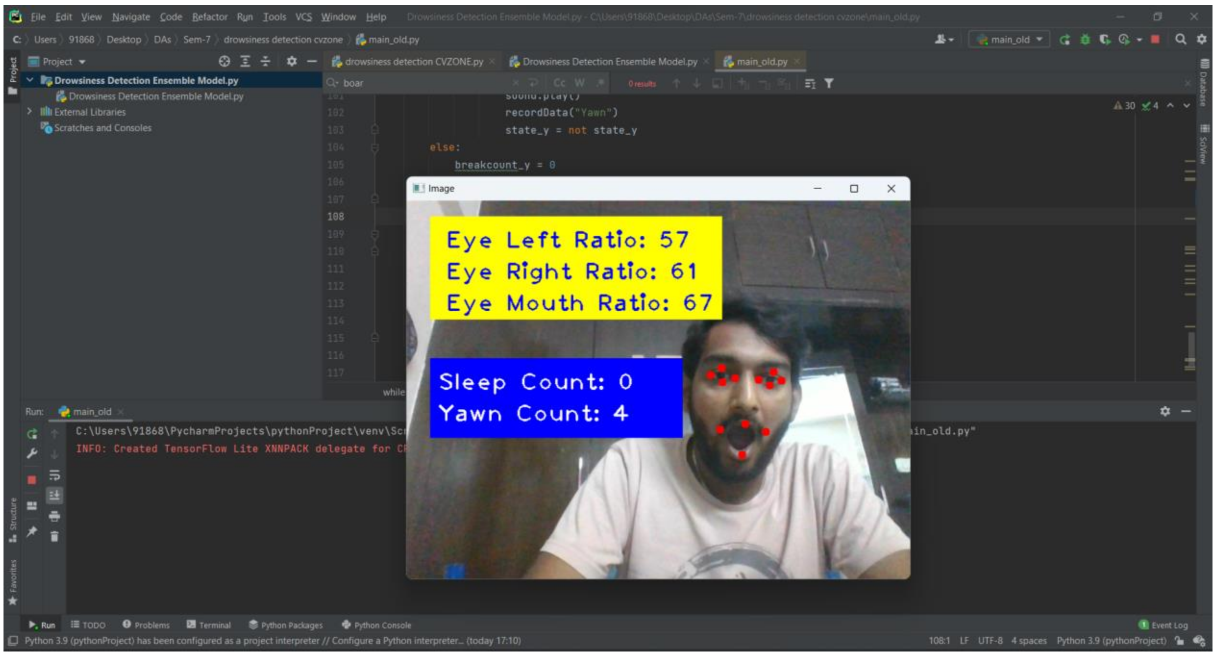
```
C:\Users\91868\PycharmProjects\pythonProject\venv\Sc                                    in_old.py"
INFO: Created TensorFlow Lite XNNPACK delegate for C
```

Eye Left Ratio: 51
Eye Right Ratio: 52
Eye Mouth Ratio: 19

Sleep Count: 0
Yawn Count: 0

File Edit View Navigate Code Refactor Run Tools VCS Window Help
Drowsiness Detection Ensemble Model.py - C:\Users\91868\Desktop\DAs\Sem-7\drowsiness detection cvzone\main_old.py
drowsiness detection CVZONE.py
Drowsiness Detection Ensemble Model.py
main_old.py
Image
Run
TODO
Problems
Terminal
Python Packages
Python Console
Event Log

Fig: Non- Drowsy/Awake since Avg. EAR > 50 & MAR < 60



```
                sound.play()
                recordData("Yawn")
                state_y = not state_y
        else:
            breakcount_y = 0
```

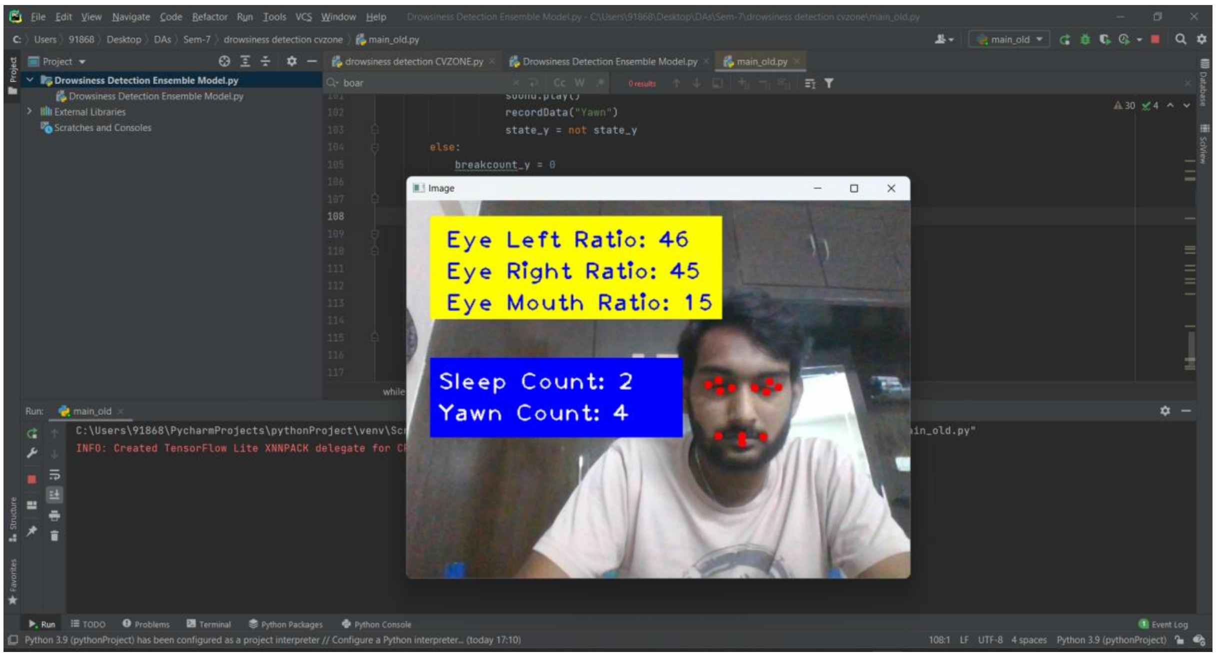
```
C:\Users\91868\PycharmProjects\pythonProject\venv\Sc                                    in_old.py"
INFO: Created TensorFlow Lite XNNPACK delegate for C
```

Eye Left Ratio: 57
Eye Right Ratio: 61
Eye Mouth Ratio: 67

Sleep Count: 0
Yawn Count: 4

File Edit View Navigate Code Refactor Run Tools VCS Window Help
Drowsiness Detection Ensemble Model.py - C:\Users\91868\Desktop\DAs\Sem-7\drowsiness detection cvzone\main_old.py
drowsiness detection CVZONE.py
Drowsiness Detection Ensemble Model.py
main_old.py
Image
Run
TODO
Problems
Terminal
Python Packages
Python Console
Event Log

Fig: Drowsy since MAR > 60, Indicating Yawning

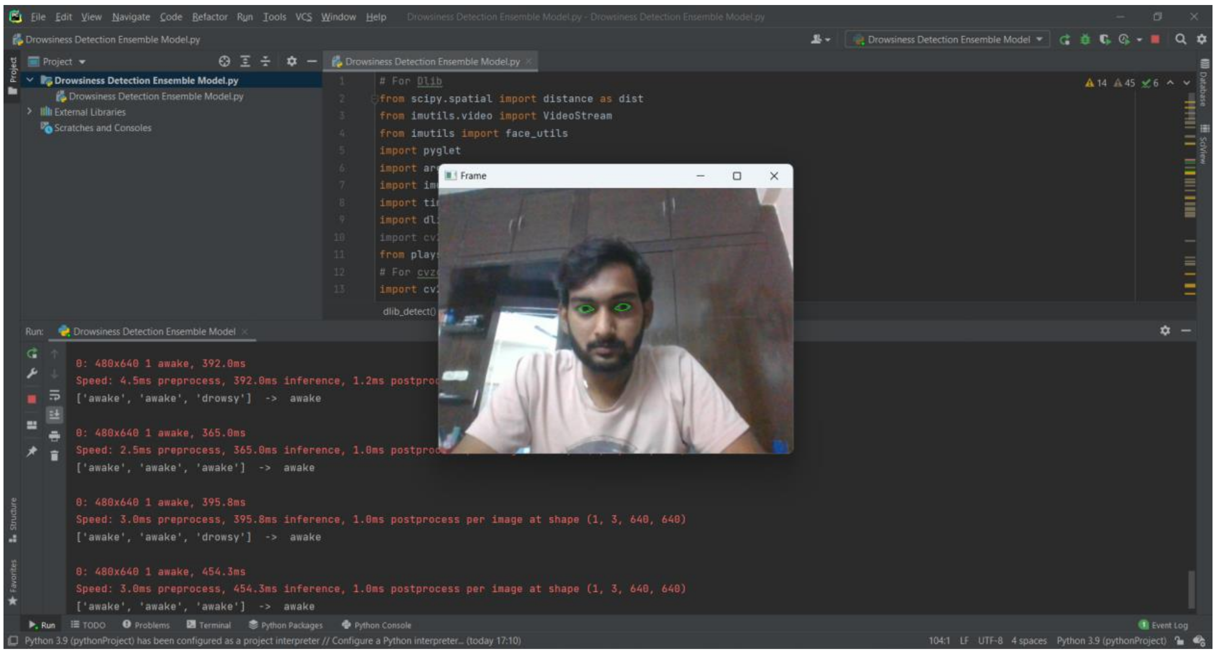


```
            sound.play()
            recordData("Yawn")
            state_y = not state_y
    else:
        breakcount_y = 0
```

```
C:\Users\91868\PycharmProjects\pythonProject\venv\Sc                    in_old.py"
INFO: Created TensorFlow Lite XNNPACK delegate for C
```

Fig: Drowsy since Avg. EAR < 50, Indicating Eyelids Closing

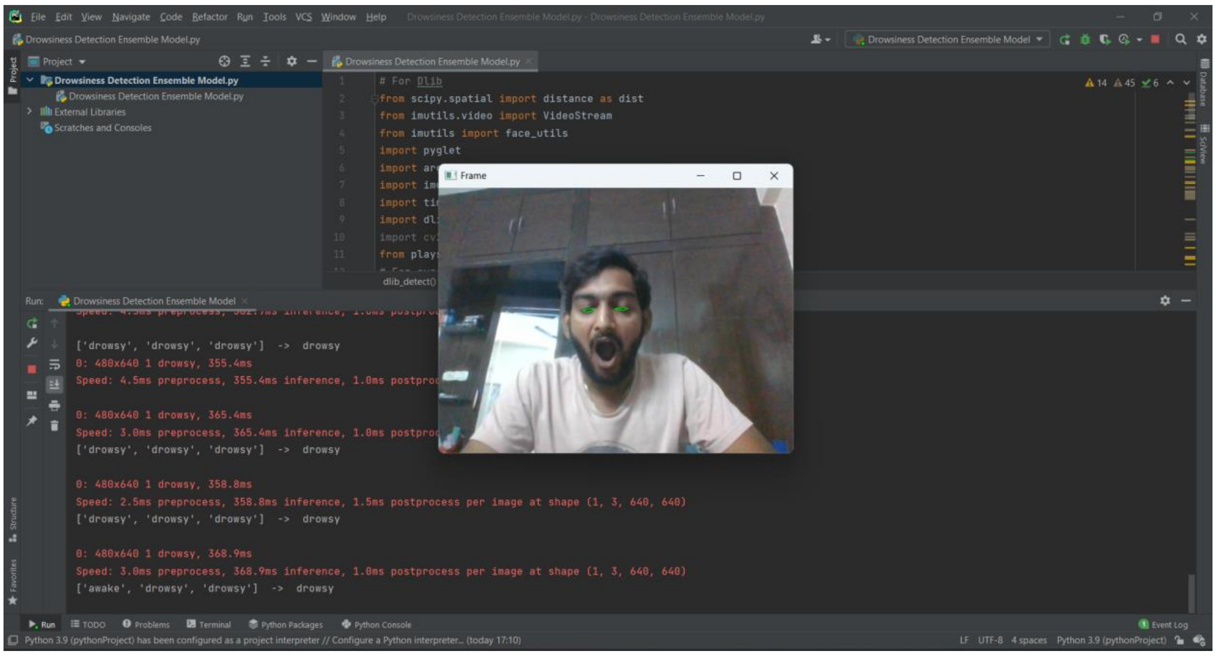
Human Fatigue Detection using Ensemble Model (Dlib, cvzone & YOLOv8):



```
# For Dlib
from scipy.spatial import distance as dist
from imutils.video import VideoStream
from imutils import face_utils
import pyglet
import ar
import im
import ti
import dl
import cv
from play
# For cvz
import cv
```

```
0: 480x640 1 awake, 392.0ms
Speed: 4.5ms preprocess, 392.0ms inference, 1.2ms postproc
['awake', 'awake', 'drowsy']  ->  awake

0: 480x640 1 awake, 365.0ms
Speed: 2.5ms preprocess, 365.0ms inference, 1.0ms postproc
['awake', 'awake', 'awake']  ->  awake

0: 480x640 1 awake, 395.8ms
Speed: 3.0ms preprocess, 395.8ms inference, 1.0ms postprocess per image at shape (1, 3, 640, 640)
['awake', 'awake', 'drowsy']  ->  awake

0: 480x640 1 awake, 454.3ms
Speed: 3.0ms preprocess, 454.3ms inference, 1.0ms postprocess per image at shape (1, 3, 640, 640)
['awake', 'awake', 'awake']  ->  awake
```

Fig: Non-Drowsy/Awake

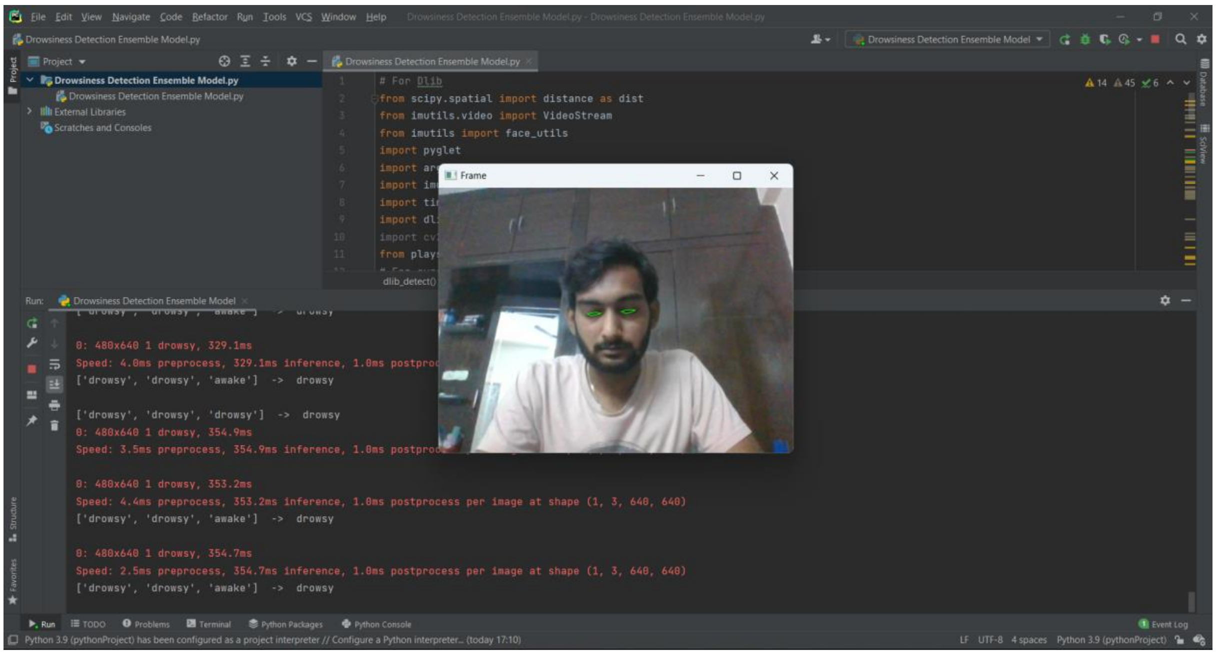


```
# For Dlib
from scipy.spatial import distance as dist
from imutils.video import VideoStream
from imutils import face_utils
import pyglet
import ar
import im
import ti
import dl
import cv
from plays
```

```
['drowsy', 'drowsy', 'drowsy']  ->  drowsy
0: 480x640 1 drowsy, 355.4ms
Speed: 4.5ms preprocess, 355.4ms inference, 1.0ms postproc

0: 480x640 1 drowsy, 365.4ms
Speed: 3.0ms preprocess, 365.4ms inference, 1.0ms postproc
['drowsy', 'drowsy', 'drowsy']  ->  drowsy

0: 480x640 1 drowsy, 358.8ms
Speed: 2.5ms preprocess, 358.8ms inference, 1.5ms postprocess per image at shape (1, 3, 640, 640)
['drowsy', 'drowsy', 'drowsy']  ->  drowsy

0: 480x640 1 drowsy, 368.9ms
Speed: 3.0ms preprocess, 368.9ms inference, 1.0ms postprocess per image at shape (1, 3, 640, 640)
['awake', 'drowsy', 'drowsy']  ->  drowsy
```

Python 3.9 (pythonProject) has been configured as a project interpreter // Configure a Python interpreter... (today 17:10)

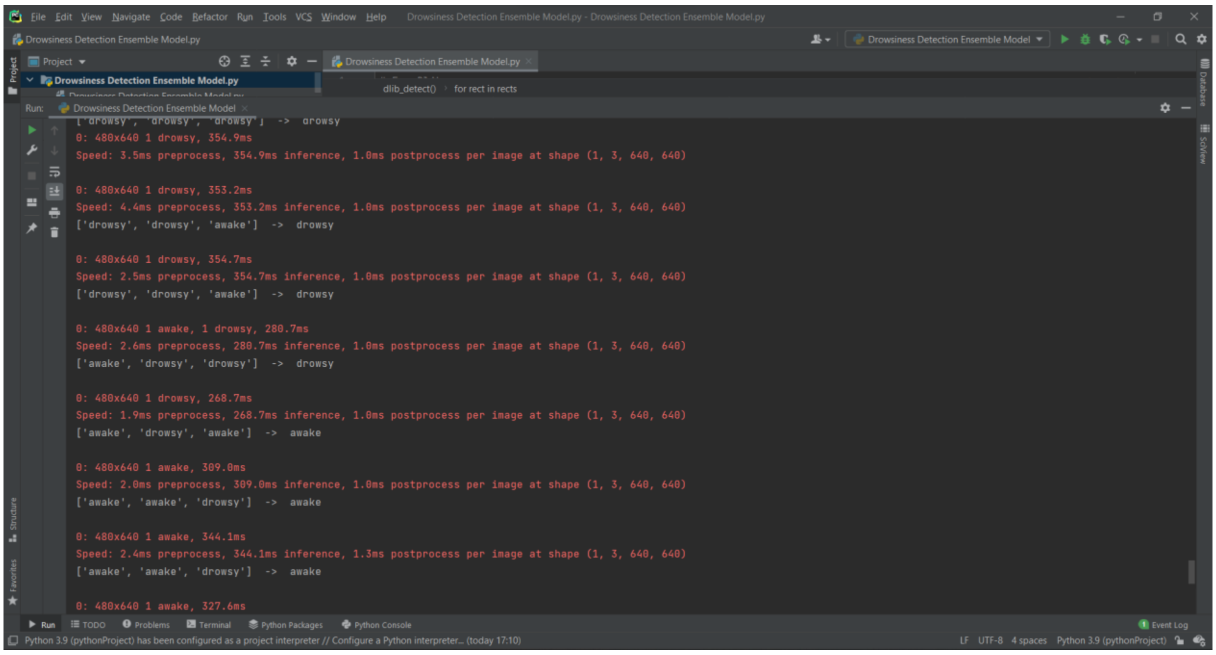
Fig: Drowsy due to yawning



```
0: 480x640 1 drowsy, 329.1ms
Speed: 4.0ms preprocess, 329.1ms inference, 1.0ms postproc
['drowsy', 'drowsy', 'awake']  ->  drowsy

['drowsy', 'drowsy', 'drowsy']  ->  drowsy
0: 480x640 1 drowsy, 354.9ms
Speed: 3.5ms preprocess, 354.9ms inference, 1.0ms postproc

0: 480x640 1 drowsy, 353.2ms
Speed: 4.4ms preprocess, 353.2ms inference, 1.0ms postprocess per image at shape (1, 3, 640, 640)
['drowsy', 'drowsy', 'awake']  ->  drowsy

0: 480x640 1 drowsy, 354.7ms
Speed: 2.5ms preprocess, 354.7ms inference, 1.0ms postprocess per image at shape (1, 3, 640, 640)
['drowsy', 'drowsy', 'awake']  ->  drowsy
```

Python 3.9 (pythonProject) has been configured as a project interpreter // Configure a Python interpreter... (today 17:10)

Fig: Drowsy due to Eyelids Closing



```
['drowsy', 'drowsy', 'drowsy']  ->  drowsy
0: 480x640 1 drowsy, 354.9ms
Speed: 3.5ms preprocess, 354.9ms inference, 1.0ms postprocess per image at shape (1, 3, 640, 640)

0: 480x640 1 drowsy, 353.2ms
Speed: 4.4ms preprocess, 353.2ms inference, 1.0ms postprocess per image at shape (1, 3, 640, 640)
['drowsy', 'drowsy', 'awake']  ->  drowsy

0: 480x640 1 drowsy, 354.7ms
Speed: 2.5ms preprocess, 354.7ms inference, 1.0ms postprocess per image at shape (1, 3, 640, 640)
['drowsy', 'drowsy', 'awake']  ->  drowsy

0: 480x640 1 awake, 1 drowsy, 280.7ms
Speed: 2.6ms preprocess, 280.7ms inference, 1.0ms postprocess per image at shape (1, 3, 640, 640)
['awake', 'drowsy', 'drowsy']  ->  drowsy

0: 480x640 1 drowsy, 268.7ms
Speed: 1.9ms preprocess, 268.7ms inference, 1.0ms postprocess per image at shape (1, 3, 640, 640)
['awake', 'drowsy', 'awake']  ->  awake

0: 480x640 1 awake, 309.0ms
Speed: 2.0ms preprocess, 309.0ms inference, 1.0ms postprocess per image at shape (1, 3, 640, 640)
['awake', 'awake', 'drowsy']  ->  awake

0: 480x640 1 awake, 344.1ms
Speed: 2.4ms preprocess, 344.1ms inference, 1.3ms postprocess per image at shape (1, 3, 640, 640)
['awake', 'awake', 'drowsy']  ->  awake

0: 480x640 1 awake, 327.6ms
```

Python 3.9 (pythonProject) has been configured as a project interpreter // Configure a Python interpreter... (today 17:10)

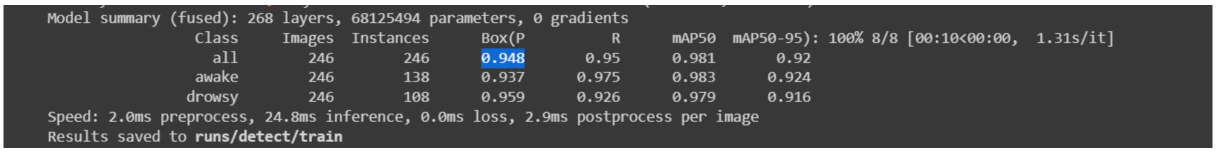
File
Edit
View
Navigate
Code
Refactor
Run
Tools
VCS
Window
Help
Drowsiness Detection Ensemble Model.py - Drowsiness Detection Ensemble Model.py
Project
Drowsiness Detection Ensemble Model.py
dlib_detect()
for rect in rects
Run:
Drowsiness Detection Ensemble Model
Run
TODO
Problems
Terminal
Python Packages
Python Console
Event Log
LF
UTF-8
4 spaces
Python 3.9 (pythonProject)

Fig: While Executing the Ensemble model, for each frame of the Footage the Drowsiness Prediction of the 3 models is printed along with the majority Prediction in the Output Box

❖ The Accuracy of Dlib for Drowsiness Detection is 78.50%. [11]

(https://www.ijert.org/research/driver-drowsiness-detection-system-IJERTV10IS120142.pdf)

❖ The Accuracy of YOLOv8 for Drowsiness Detection is 94.80%.



```
Model summary (fused): 268 layers, 68125494 parameters, 0 gradients
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 8/8 [00:10<00:00,  1.31s/it]
                   all        246        246      0.948       0.95      0.981       0.92
                 awake        246        138      0.937      0.975      0.983      0.924
                drowsy        246        108      0.959      0.926      0.979      0.916
Speed: 2.0ms preprocess, 24.8ms inference, 0.0ms loss, 2.9ms postprocess per image
Results saved to runs/detect/train
```

❖ The Accuracy of cvzone for Drowsiness Detection is 95.99%. [12]

(http://www.journal-aquaticscience.com/article_136033_c96172a0f080d21c810d21c25036952d.pdf)

❖ When we tested the above Ensemble model, using a 200-photo dataset of different colors & lighting conditions, our ensemble model gave the correct output for 197 Images. So, as the prototype we built worked 197/200 times correctly, the accuracy of this System will result as “98.50%”.

❖ Hence, the Ensemble model we created resulted in more accuracy than all the 3 models when executed individually. So, Ensembling the results of all the 3 models using majority voting made our system highly accurate & reliable.

6. CONCLUSION

In this project, we successfully developed a software & hardware prototype for intelligent 360-degree safety for smart cars, focusing on two crucial aspects: drowsiness detection and alcohol detection respectively. Through extensive testing and evaluation, we have achieved promising results and demonstrated the effectiveness of our system.

Firstly, we conducted testing on our hardware prototype of the Alcohol Detection Module by manipulating the proximity of alcohol to the MQ-3 sensor approximately 150 times. The prototype provided accurate results in 146 instances, demonstrating a success rate of **97.33%**. Although there were a few instances of significant delays, the overall performance of the system was highly reliable.

For drowsiness detection, we evaluated the individual performance of three models: Dlib, YOLOv8, and cvzone. According to previous research studies, Dlib achieved an accuracy of 78.50%, YOLOv8 achieved 94.80% accuracy (Got this Result while Training), and cvzone achieved 95.99% accuracy. We further improved the accuracy by ensembling the results of all three models using majority voting. When tested with a dataset of 200 photos under various lighting conditions, our ensemble model produced accurate results for 197 images, resulting in an impressive accuracy rate of **98.50%**. The ensembling technique proved to be highly effective in enhancing the overall accuracy and reliability of the system.

Overall, our prototype demonstrated excellent performance in both alcohol detection and drowsiness detection, showcasing accuracy rates of **'97.33%'** & **'98.50%'**, respectively. These results highlight the potential of our system to significantly enhance road safety by alerting drivers to the presence of alcohol and detecting drowsiness in real-time.

Table-1: Accuracy Table

Model		**Accuracy**
Alcohol Detection Module		"97.33%"
Human Fatigue Detection Module (Ensemble Model)		"98.50%"
Individual Performance(s)	Dlib	78.50%
	YOLOv8	94.80%
	Cvzone	95.99%

Moving forward, there are several avenues for future improvement and expansion of the project. Firstly, we can explore the integration of additional sensors and algorithms to enhance the detection capabilities of the system. For example, incorporating sensors to detect other impairments, such as drug intoxication or distraction, could further improve the safety features of the system. Furthermore, the development of a comprehensive monitoring and reporting system would enable long-term analysis and data-driven insights. By collecting and analyzing data on driver behavior and patterns, we can identify trends, develop predictive models, and provide personalized feedback to individuals, promoting safer driving practices.

In conclusion, our project has successfully developed an intelligent 360-degree safety system for smart cars, incorporating drowsiness detection and alcohol detection. With high accuracy rates and potential for future enhancements, this system holds great promise in preventing accidents and promoting safer driving practices.

7. REFERENCES

Journal:

[1]. M. Abdelsalam and T. Bonny, "IoV Road Safety: Vehicle Speed Limiting System," 2019 International Conference on Communications, Signal Processing, and their Applications (ICCSPA), Sharjah, United Arab Emirates, 2019, pp. 1-6, doi: 10.1109/ICCSPA.2019.8713713.

[2]. Sharanabasappa, J. N. Sayed Farooq, V. N. Soundarya, V. S. Rao and K. S. Chandraprabha, "Safe Drive: An Automatic Engine Locking System to Prevent Drunken Driving," 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India, 2018, pp. 1957-1961, doi: 10.1109/RTEICT42901.2018.9012404.

[3]. R. R. Varghese, P. M. Jacob, J. Jacob, M. N. Babu, R. Ravikanth and S. M. George, "An Integrated Framework for Driver Drowsiness Detection and Alcohol Intoxication using Machine Learning," 2021 International Conference on Data Analytics for Business and Industry (ICDABI), Sakheer, Bahrain, 2021, pp. 531-536, doi: 10.1109/ICDABI53623.2021.9655979.

[4]. M. Siwach, S. Mann and D. Gupta, "A Practical Implementation of Driver Drowsiness Detection Using Facial Landmarks," 2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 2022, pp. 1-4, doi: 10.1109/ICRITO56286.2022.9964990.

[5]. K. Jose, S.M. Raj, A. S, S. M, and V. Surendran, "Alcohol Sensing Alert with Engine Locking," International Journal of Innovative Research in Science, Engineering and Technology, vol. 8, no. 6, pp. 19-22, June 2021, ISSN-2349-5162.

[6]. Fouzia, R. Roopalakshmi, J. A. Rathod, A. S. Shetty and K. Supriya, "Driver Drowsiness Detection System Based on Visual Features," 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, India, 2018, pp. 1344-1347, doi: 10.1109/ICICCT.2018.8473203.

[7]. A. Reddy K., S. Patel, K. P. Bharath and R. Kumar M., "Embedded Vehicle Speed Control and Over-Speed Violation Alert Using IoT," 2019 Innovations in Power and Advanced Computing Technologies (i-PACT), Vellore, India, 2019, pp. 1-5, doi: 10.1109/i-PACT44901.2019.8960008.

[8]. H. C. T., S. Nagaraju, B. N. Varma, K. K. M., M. S. and M. K. V., "IoT based Vehicle Over-Speed Detection and Accident Avoidance System," 2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON), Bengaluru, India, 2021, pp. 112-117, doi: 10.1109/CENTCON52345.2021.9688241.

[9]. A. Ghosh and R. Prasad, "Monitoring of Road Accidents- A Review," 2018 International Conference on Control, Power, Communication and Computing Technologies (ICCPCCT), Kannur, India, 2018, pp. 7-12, doi: 10.1109/ICCPCCT.2018.8574276.

[10]. C. M. Sunny et al., "Forecasting of Road Accident in Kerala: A Case Study," 2018 International Conference on Data Science and Engineering (ICDSE), Kochi, India, 2018, pp. 1-5, doi: 10.1109/ICDSE.2018.8527825.

[11]. V. Iyer, A. Vanjari and V. Patil, "Driver Drowsiness Detection System," International Journal of Engineering Research & Technology (IJERT), India, 2021, pp. 275-278.

[12]. L.Thulasimani, Poojeevan P, and Prithashasni S P, " Real Time Driver Drowsiness Detection Using Opencv And Facial Landmarks," International Journal of Aquatic Science, India, 2021, pp. 4297-4314.

APPENDIX A

Human Fatigue Detection Code:

```
# For Dlib
from scipy.spatial import distance as dist
from imutils.video import VideoStream
from imutils import face_utils
import pyglet
import argparse
import imutils
import time
import dlib
import cv2
from playsound import playsound
# For cvzone
import cv2
from cvzone.FaceMeshModule import FaceMeshDetector
import pyfirmata
# For YOLO
from ultralytics import YOLO


def sound_alarm(path):
    # play an alarm sound
    music = pyglet.resource.media('alarm.wav')
    music.play()
    pyglet.app.run()

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

    # compute the euclidean distance between the horizon
    # eye landmark (x, y)-coordinates
    C = dist.euclidean(eye[0], eye[3])

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

    # return the eye aspect ratio
    return ear

def dlib_detect(frame):
    EYE_AR_THRESH = 0.3

    # 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"]

    frame = imutils.resize(frame, width=450)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # detect faces in the grayscale frame
    rects = detector(gray, 0)

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

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

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

        #cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
        # 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:
            return "drowsy",frame
        else:
            return "awake",frame
    return "awake",frame


def yolo_detect(frame):
    results = model.predict(frame)
    result = results[0]
    for box in result.boxes:
        class_id = result.names[box.cls[0].item()]
        if(class_id=="drowsy"):
            return "drowsy"
        else:
            return "awake"


def cvzone_detect(frame):
    img = cv2.flip(frame, 1)
    img, faces = detector1.findFaceMesh(img, draw=False)

    if faces:
        face = faces[0]
        eyeLeft = [27, 23, 130, 243]  # up, down, left, right
        eyeRight = [257, 253, 463, 359]  # up, down, left, right
        mouth = [11, 16, 57, 287]  # up, down, left, right

        #calculate eye left distance ratio
        eyeLeft_ver, _ = detector1.findDistance(face[eyeLeft[0]],
face[eyeLeft[1]])
        eyeLeft_hor, _ = detector1.findDistance(face[eyeLeft[2]],
face[eyeLeft[3]])
        eyeLeft_ratio = int((eyeLeft_ver/eyeLeft_hor)*100)
        # calculate eye right distance ratio
        eyeRight_ver, _ = detector1.findDistance(face[eyeRight[0]],
face[eyeRight[1]])
        eyeRight_hor, _ = detector1.findDistance(face[eyeRight[2]],
face[eyeRight[3]])
        eyeRight_ratio = int((eyeRight_ver / eyeRight_hor) * 100)
        # calculate mouth distance ratio
        mouth_ver, _ = detector1.findDistance(face[mouth[0]], face[mouth[1]])
        mouth_hor, _ = detector1.findDistance(face[mouth[2]], face[mouth[3]])
        mouth_ratio = int((mouth_ver / mouth_hor) * 100)

        # drowsiness detection logic
        if (mouth_ratio > 60) or (eyeLeft_ratio <= 50) or (eyeRight_ratio <=
50):
            return "drowsy"
        else:
            return "awake"
    else:
        return "awake"


detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor("68_face_landmarks.dat")

model = YOLO("best.pt")

detector1 = FaceMeshDetector(maxFaces=1)

EYE_AR_CONSEC_FRAMES = 15
ALARM_ON = False
COUNTER = 0
```

```
output=["temp","temp","temp"]

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-w", "--webcam", type=int, default=0, help="index of webcam
on system")
args = vars(ap.parse_args())
# start the video stream thread
time.sleep(1)
vs = VideoStream(src=args["webcam"]).start()

# 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()
    output[0],frame = dlib_detect(frame)
    output[1] = yolo_detect(frame)
    output[2] = cvzone_detect(frame)
    majority_element = max(output, key=output.count)
    print(output," -> ",majority_element)

    ########## Aakash
    if(majority_element=="drowsy"):
        # if the eyes were closed for a sufficient number of
        # then sound the alarm
        COUNTER = COUNTER + 1
        if COUNTER >= EYE_AR_CONSEC_FRAMES:
            # if the alarm is not on, turn it on
            if not ALARM_ON:
                ALARM_ON = True
            # draw an alarm on the frame
            cv2.putText(frame, "DROWSINESS ALERT!", (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
            playsound('alarm.wav')

    # otherwise, the eye aspect ratio is not below the blink
    # threshold, so reset the counter and alarm
    else:
        COUNTER = 0
        ALARM_ON = False
    # 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
# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```

Alcohol Detection Code:

```
/*
  Alcohol Detection
*/

#include<LiquidCrystal.h>

#define buzzer 2
#define LED 3
#define sensorAnalog A1
#define button 8
#define LED2 13
#define sensorAnalog A1 //MQ-3

int limit = 400;
int count = 0;
int timer = 20;
int emergency = false;

LiquidCrystal lcd(12, 11, 7, 6, 5, 4);

void setup()
{
  pinMode(sensorAnalog, INPUT);
  pinMode(LED, OUTPUT);
  pinMode(buzzer, OUTPUT);
  lcd.begin(16, 2);
  pinMode(LED2, OUTPUT);
  pinMode(button, INPUT);
  pinMode(sensorAnalog, INPUT);
  Serial.begin(9600);
}

void loop()
{
  int analog = analogRead(sensorAnalog);
  //int ALCOHOL_detected=1;
  Serial.print("Analog value : ");
  Serial.println(analog);

  if(emergency==true)
  {
    digitalWrite(LED2, HIGH);
    lcd.setCursor(1,0);
    lcd.print("Emergency Mode");
    lcd.setCursor(3,1);
    lcd.print("Activated!");
  }
  else if(analog>=650)
  {
    count = count + 1;
    tone(buzzer, 650);
    digitalWrite(LED, HIGH);
    delay(250);
    noTone(buzzer);
    digitalWrite(LED, LOW);
    delay(250);
  }
  else
  {
    count = 0;
    timer = 20;
    lcd.setCursor(0,0);
    lcd.print("                ");
    lcd.setCursor(0,1);
    lcd.print("                ");
    emergency = false;
    digitalWrite(LED2, LOW);
    digitalWrite(LED, LOW);
    digitalWrite(buzzer, LOW);
  }

  if(count>=40 && count%2==0 && count<80 && emergency==false)
  {
    lcd.setCursor(0,0);
    lcd.print("Emergency Mode??");
    lcd.setCursor(5,1);
    lcd.print("00:");
    if(timer>=10)
    {
      lcd.setCursor(8,1);
      lcd.print(timer);
    }
    else
    {
      lcd.setCursor(8,1);
      lcd.print("0");
      lcd.setCursor(9,1);
      lcd.print(timer);
    }
    timer = timer - 1;
  }

  if(count>=40 && count<80)
  {
    if (digitalRead(button) == HIGH)
    {
      emergency = true;
      digitalWrite(LED2, HIGH);
      lcd.setCursor(0,0);
      lcd.print("                ");
      lcd.setCursor(0,1);
      lcd.print("                ");
      lcd.setCursor(1,0);
      lcd.print("Emergency Mode");
      lcd.setCursor(3,1);
      lcd.print("Activated!");
    }
    while(digitalRead(button) == true);
    delay(50);
  }

  if(count==80)
  {
    lcd.setCursor(0,0);
    lcd.print("                ");
    lcd.setCursor(0,1);
    lcd.print("                ");
    lcd.setCursor(5,0);
    lcd.print("Danger");
    lcd.setCursor(1,1);
    lcd.print("Engine Locked!");
  }


}
```