CHAPTER 1

INTRODUCTION

1.1 OVERVIEW:

This study presents an IoT based smart helmet system with frequency based vehicle control designed to augment road safety through an integrated approach. The system interfaces seamlessly with an ARDUINO UNO microcontroller, utilizing a multifaceted modules affixed to the helmet.

This innovative project focuses on revolutionizing driver safety through the development of a comprehensive smart system. The system is designed to address multiple safety concerns, including drowsiness detection, accident response, and blindspot monitoring. In the helmet section, the project integrates the LeNet Convolutional Neural Network (CNN) architecture, a pioneering deep learning model, for real-time drowsiness detection. LeNet analyzes facial features to promptly identify signs of driver fatigue, ensuring immediate alerts and proactive intervention.

Simultaneously, the vehicle section incorporates a vibration sensor strategically positioned to detect sudden impacts, serving as a crucial component for accident detection. This sensor acts as a rapid-response mechanism, triggering emergency notifications in the event of an impact. Moreover, the project introduces a sophisticated blindspot detection system with sensors covering a 180-degree area behind the driver. This system plays a pivotal role in identifying potential collisions in blindspots and notifies the driver well in advance, offering a preventive measure to avoid accidents.

The envisioned smart safety system aims to provide a holistic approach to driver safety, enhancing awareness, mitigating potential dangers, and facilitating immediate responses to ensure a safer driving experience. Through the integration of cutting-edge technologies and a multi-faceted approach, the project aspires to set new standards in proactive driver safety systems.

1.2 DESCRIPTION:

Introducing an advanced smart helmet system designed to elevate road safety through an integrated and sophisticated approach. Seamlessly interfacing with an ARDUINO UNO microcontroller, the helmet module incorporates a push button that mandates proper helmet usage by the driver and upon wearing the helmet the bike will start. The main aim of the project is to elevate road safety and driver’s safety by holistic approach such as drowsiness monitoring and blind spots monitoring which will alert the driver and protect them from road hazards.

Zigbee transmitter and receiver units facilitate seamless communication between the helmet and the vehicle. Accident detection using a vibration sensor prompts the GSM module to send an immediate alert, and the GPS module transmits the accident location. An ultrasonic sensor, coupled with a servo motor for rotational monitoring, automatically halts the vehicle upon obstacle identification.

This comprehensive smart helmet ensures holistic road safety by addressing proper helmet usage, alcohol detection, distracted driving prevention, real-time accident reporting, and obstacle avoidance through a synergy of sensors and communication modules, significantly minimizing road accidents and enhancing overall driving safety.

1.3 OBJECTIVE:

The central objective of this pioneering project is to elevate driver safety through the development and implementation of a comprehensive smart system. The primary focus is on real-time drowsiness detection, leveraging the LeNet Convolutional Neural Network architecture to analyze facial features promptly and issue immediate alerts in response to signs of driver fatigue. Simultaneously, the project aims to establish a robust accident response system by integrating a vibration sensor in the vehicle section, enabling swift detection of accidents and triggering emergency notifications in the event of sudden impacts. Furthermore, the project endeavors to enhance blindspot monitoring by implementing a sophisticated detection system covering a 180-degree area behind the driver, providing advanced warnings to mitigate collision risks. Addressing challenges in optimizing the LeNet model and seamlessly integrating data from various sensors, including the vibration sensor and blindspot detection system, is a key aspect of the project. Additionally, ensuring the reliable operation of communication modules, including GPS and GSM, is crucial for accurate emergency notifications and effective communication between the helmet and the vehicle. Ultimately, the project strives to establish new standards in proactive driver safety, contributing to overall driver awareness and fostering a safer and more attentive driving experience.

1.4 ABOUT THE PROJECT:

This innovative project aims to revolutionize driver safety through a comprehensive smart system. The LeNet Convolutional Neural Network is employed in the helmet for real-time drowsiness detection, ensuring prompt alerts. Simultaneously, the vehicle section incorporates a vibration sensor for rapid accident detection, and a 180-degree blindspot detection system provides advanced warnings. Challenges include optimizing the LeNet model and integrating data seamlessly. The system strives to enhance overall driver safety by addressing drowsiness, accidents, and blind spots, setting new standards in proactive safety systems through cutting-edge technology integration.

1.5 STRUCTURE OF THE REPORT:

Chapter 1 presents a smart helmet system aimed at bolstering road safety. Integrated with an ARDUINO UNO microcontroller, the helmet module incorporates sensors and communication units to tackle challenges like helmet adherence, alcohol detection, distracted driving, real-time accident reporting, and obstacle avoidance. The project's core objective is to significantly enhance driving safety through a holistic approach.

Chapter 2 conducts a concise survey of relevant studies, including Arduino-based smart helmets, accident detection systems, and IoT implementations. Each study contributes valuable insights, shaping the groundwork for the current project by exploring various methodologies and technological advancements in the domain of smart helmets.

Chapter 3 outlines the challenges addressed by the smart helmet project and reviews the limitations of existing systems. It introduces the proposed system, emphasizing the integration of sensors and communication modules. The block diagram visually represents the architecture, setting the stage for subsequent chapters on design and implementation.

Chapter 4 outlines the design process of the smart helmet, focusing on module integration and architectural considerations. It details how various sensors, such as the Zigbee transmitter, are seamlessly integrated into the helmet. The architectural diagram provides a visual representation of the system's structure, emphasizing the relationships between different components.

Chapter 5 delves into the practical implementation of the smart helmet using the Arduino IDE and embedded C. It covers the utilization of sensors mentioned in the design phase, illustrating how the IoT-based system is developed and programmed. The chapter provides insights into the technical aspects, code implementation, and the role of different sensors in making the smart helmet operational.

Chapter 6, results and analysis, presents the results of the Proposed system and the analysis of the Smart helmet systems, including the vehicle section and generate the accuracy of the systems.

Chapter 7, concludes the project by summarizing the key findings and outcomes. It reflects on the overall success of the smart helmet in addressing road safety issues. Additionally, the chapter explores potential avenues for future enhancements, suggesting areas for further development or improvement in the smart helmet system to adapt to evolving technological and safety standards.

CHAPTER 2

LITERATURE SURVEY

Mahesh S Gour(Dept of ECE),Druva Kumar S(Dept of ECE),pradeep Kumara(Dept of VLSI design),Majuthana S(Dept of VLSI design) in **Arduino based smart and intelligent helmet system for two-wheelers** [1] proposed an smart helmet system is designed for enhanced road safety, integrating various modules with the primary Arduino processor. Two modules, one on the helmet and the other on the bike, utilize RF transmitters and receivers for synchronization. When the helmet is worn, encrypted data is transmitted to the bike module, enabling the ignition only when validation conditions, like helmet usage, are met. An alcohol detection sensor serves as a breathalyzer, preventing bike ignition if alcohol is detected. In the event of an impact, a vibration sensor triggers GSM and GPS modules, sending an SMS with GPS coordinates to the registered number and activating a buzzer for attention. The helmet includes a microphone and speaker for emergency communication. This comprehensive system addresses helmet usage, alcohol detection, distracted driving prevention, real-time accident reporting, and obstacle avoidance through sensor integration and communication modules.

Pranav Pathak(Electrical Engineering dept) in his **Smart Helmet with Motorbike unit for Accident and Rash Driving Detection**[2] project proposes a system that comprises two units: the Helmet Unit and the Motorbike Unit (MU), communicating via RF using the NRF24L01 module. The Helmet Unit, equipped with various sensors embedded in an Arduino Nano, includes a pulse rate sensor to measure the rider's pulse rate. Upon wearing the helmet and locking the buckle, the NRF module activates, establishing a connection with the Motorbike Unit. The pulse rate data is continuously transmitted to the MU. The MU features a LIDAR sensor detecting vehicles behind the bike, alerting the rider to approaching vehicles for enhanced awareness.Additionally, sensors on the handle grip and middle of the bike detect the position of the rider's hands and legs. The system alerts the rider if they assume a risky riding position. The Helmet Unit, powered by Arduino Nano with ATmega328, is lightweight due to the compact size of the board and sensors. The helmet includes a pulse rate sensor and an MQ-3 alcohol sensor on the front side. The Motorbike Unit, utilizing an Arduino UNO , receives sensor data from the Helmet Unit via the NRF module. It features various sensors, including a hall effect sensor for bike speed, ultrasonic and infrared sensors for hand and leg position detection, GSM and GPS modules for messaging with rider location, an LCD screen for displaying messages, a buzzer for rider alerts, a LIDAR sensor for identifying approaching vehicles, a push-button switch for rider safety, a relay for ignition control, and an accelerometer for measuring bike tilt and accident identification.

Jesudoss A,Vyabhavi R and Anusha B in **Design of Smart Helmet for Accident Avoidance** [3] proposed a smart helmet integrates multiple sensors, including an IR sensor, load sensor, vibration sensor, gas sensor for alcohol detection, and MEMS technology for handlebar control. It aims to enhance safety by monitoring parameters such as vehicle load, potential accidents through vibrations, and detecting alcohol consumption. The system requires users to create accounts with the server, storing essential details for user access control. Load monitoring utilizes a load cell, issuing alerts if the load surpasses a specified limit. Alcohol detection relies on a gas sensor near the helmet's mouth area, issuing warnings for drunk driving. The microcontroller serves as the system's core, facilitating communication with the server and ensuring an automated safety system for vehicles. Accident detection involves a vibration sensor identifying sudden impacts, triggering alerts to the police. The system continuously monitors accelerometer output and external impact sensors, taking appropriate measures in case of severe accidents, including notifying hospitals.

Huma Afreen, jhuita Nandi, Rohit Kundu, Ankit Dutta in **SMART HELMET FOR MOTORCYCLES** [4].Their project serves multiple critical objectives aimed at enhancing rider safety. Firstly, the system promotes responsible riding habits by ensuring the motor starts only when the rider wears the helmet and activates the switch. This mechanism encourages compliance with essential safety practices. Additionally, the project incorporates an alcohol content testing feature using an MQ-3 sensor. While programmed for demonstration purposes, the system sets a threshold limit of 0.04 mg/L, below the legal limit for driving under the influence. If the alcohol content exceeds this limit, the system prevents the rider from operating the motorcycle, addressing a key safety concern.Furthermore, the system integrates a vibration sensor for accident detection. By identifying frequencies generated during accidents or encounters with obstacles, the system can promptly alert the rider to potential hazards. In the unfortunate event of an accident, the project facilitates effective emergency response by continuously sending latitude and longitude coordinates to pre-registered SIM numbers via the GSM module. This real-time location tracking ensures swift assistance and improves overall safety measures on the road. In essence, the smart helmet project adopts a holistic approach to rider safety, addressing issues of helmet usage, alcohol consumption, accident detection, and location tracking.

J Joy Mathavan,V K D Wijesekara,N Satheeskanth, W M U J Wanasinghe,M Maathushan,V V Wijenayake(Department of Engineering Technology, Faculty of Technology,Kilinochchi Premises,Ariviyal Nagar, University of Jaffna, Sri Lanka) in **Smart helmet to start the motorbike and to prevent accidents** [5] introduces an smart helmet that is designed to start the motorbike and prevent accidents through three distinct modes: Start ON mode, Running Mode, and Accident mode. These modes utilize various sensors and have been tested in real-time environments. It aims to enhance motorcycle safety through three modes: Start ON, Running, and Accident. In the Start ON mode, the helmet's controller checks if it's worn properly, the side stand is up, and there's no alcohol consumption before activating the engine. The Running mode utilizes a gyroscopic sensor to monitor the bike's inclination, automatically stopping the engine if the bike tilts beyond a set angle, potentially signaling an accident. The Accident mode is triggered in case of a significant tilt, sending an "accident detected" message to a pre-registered phone number. The successfully tested model includes a display screen integrated into the motorcycle's dashboard for mode information and emergency messages. This comprehensive system ensures engine activation conditions, monitors bike inclination to prevent accidents, and facilitates timely communication in case of emergencies.

Guntupalli Sireesha,Anusha,K. Baby Satya Jahnavi,Ayusha Baburay(Computer science dept) in **Smart Helmet using IoT** [6] introduces an system that achieves three main objectives: monitoring the status of the rider wearing a helmet, detecting alcohol content, and detecting and sharing the location of accidents.In the helmet section, a transmitter is placed in the helmet, and a receiver is installed on the bike, enabling wireless communication between the two modules. When the rider wears the helmet, a switch is pressed, and an alcohol sensor in the helmet detects alcoholic gas from the rider's breath. If the alcohol sensor indicates an excessive alcohol level, the bike's ignition is turned off. The helmet also contains an accelerometer to measure helmet tilting, and the microcontroller embedded in the helmet processes the data. The output is transmitted to the bike module through an RF transmitter.To start the bike ignition, two conditions must be met: the rider must wear the helmet, as confirmed by the switch in the helmet, and the rider's alcohol consumption should not exceed a threshold value. The bike's ignition will only start when both conditions are satisfied. The accelerometer in the helmet monitors the helmet's tilt concerning the ground, and if it exceeds the threshold value, indicating an accident, the system immediately notifies a registered contact number using GSM. Additionally, it sends the location of the accident, allowing for prompt assistance and medical treatment.

Mohammad Ehsanul Alim (School of Electrical & Electronic Engineering),Sarosh Ahmad(Department of Electrical Engineering),Marzieh Naghdi Dorabati(Faculty of Computer Engineering),Ihab Hassoun(Faculty of Engineering) in **Design and Implementation of IOT Based Smart Helmet for Road Accident Detection** [7].The paper presents a Smart Helmet design using IoT, incorporating Arduino Mega-2560 for the bike unit and Arduino Nano for the helmet unit. The helmet features a sharp IR sensor for detecting helmet usage and an MQ-3 sensor for alcohol detection. The nRF24L01 RF module enables communication between the helmet and bike units, with an OLED display issuing warnings for drunk driving and non-helmet usage. Testing reveals that the bike only starts when the user is sober and wearing a helmet. The system integrates a GSM-GPS module for accident detection and location tracking. In case of an accident, the GPS module identifies the location, displayed on Google Maps, and sends a text message to the rider's family for prompt assistance. The comprehensive system aims to enhance rider safety by preventing drunk driving, ensuring helmet usage, and facilitating quick response in case of accidents.

Dr.S.Sekar , L. Jaivenkatesh , S.Aravind Kumar, K.DineshKumar, N.Jeevanantham (Assistant Professor (Sel.G), Department of IT,Student, Department of Information Technology) in **IOT BASED SMART HELMET** [8] proposed motorcycle safety system targets helmet non-usage, drunk driving, and delayed accident response. It consists of a Helmet Unit and a Bike Unit, employing specialized sensors and modules. An Infrared sensor in the helmet ensures proper usage, blocking ignition if worn incorrectly. A MQ-3 gas sensor detects alcohol levels, stopping ignition if alcohol is present. Radio Frequency communication connects the Helmet Unit as a transmitter and the Bike Unit as a receiver. The Bike Unit, with Arduino MEGA, Ultrasonic Sensors, GSM SIM800L, and GPS Neo 6m, verifies helmet usage and checks for alcohol. Ultrasonic sensors address blind spots, alerting the rider to approaching vehicles. In accidents, the GPS module identifies the location, and GSM sends swift SMS notifications to emergency contacts.

Mohamed A. Torad ,Mustafa Abdul Salam in **Smart helmet using internet of things** [9] proposed an smart helmet system integrates hardware and software components, utilizing Android, Firebase, and Arduino tools for its implementation. The system integrates accident detection and real-time tracking using a Piezoelectric Sensor to capture various environmental changes. An Android mobile app connects to the helmet through Firebase Realtime Database, serving as a central hub for data storage and management. Firebase's low-latency synchronization is leveraged for efficient real-time updates, making it a suitable solution. The tracking system utilizes GPS and Arduino to facilitate location sharing, emergency notifications, and tracking features. The mobile app, featuring two Java classes (PhoneAuthActivity and MapsActivity), ensures user privacy through authentication methods. An "Add friends" function streamlines permission for tracking, emphasizing privacy and mutual consent. The system's efficiency lies in the detailed algorithms for phone authentication and data retrieval from the database.

P Koteswara Rao, P Tarun Sai, N Vinay Kumar, SK Yusuf Vidya Sagar in **Design And Implementation Of Smart Helmet Using IoT** [10] introduced an smart helmet concept prioritizes rider safety by checking for helmet usage and sobriety. If both conditions are met, the ignition is allowed; otherwise, it remains off, with an LCD display indicating the status. The system incorporates a mercury switch for accident detection, triggering GPS and GSM messages to notify a designated contact person with exact coordinates. IoT integration involves storing rider data, including helmet condition, alcohol status, and accidents, on ThingSpeak. An 8-bit AVR microcontroller manages system functions, utilizing a relay module for ignition control. IoT cloud services enhance data management, offering scalability, while a DC power supply converts AC to various voltages for system components.

CHAPTER 3

PROBLEM DEFINITION AND METHODOLOGIES

3.1 PROBLEM DEFINITION:

The persistent challenges in road safety have spurred the need for an innovative smart helmet system, driven by a refined working principle. The existing landscape of safety measures grapples with issues such as improper helmet usage, alcohol-influenced driving, distracted driving, and accidents. The insufficiency of current safety systems stems from a fragmented working principle, necessitating a comprehensive solution. The crux of the problem lies in the absence of a seamlessly integrated system that harmoniously employs advanced sensors and communication modules within a helmet. Ensuring proper helmet adherence, precise alcohol detection, distraction prevention, real-time accident reporting, and efficient obstacle avoidance require a unified working principle. The challenge is to devise a smart helmet system that not only enforces safety measures but also proactively mitigates potential risks on the road through a cohesive working principle. Overcoming the existing limitations in safety technology and formulating a robust solution hinge upon the development of a smart helmet with a refined working principle. Such a system becomes integral in shaping a safer road environment, instilling responsible driving habits, and minimizing the repercussions of accidents. Crafting an innovative smart helmet system that aligns with an advanced working principle stands as a paramount challenge, addressing the evolving expectations for road safety in our technologically advancing world.

3.2 EXISTING SYSTEMS:

Traditional systems for motorcycle safety primarily center around protective gear, primarily helmets, and basic electronic aids. Standard helmets are designed with hard outer shells and a cushioned interior to absorb impact and protect the rider’s head during accidents. However, these helmets do not include technology to monitor the rider's condition or the surrounding environment. In terms of electronics, some modern helmets incorporate Bluetooth technology for communication purposes, allowing riders to answer calls hands-free or listen to music, enhancing the riding experience but not significantly improving active safety measures. Additionally, a few helmets come equipped with rear-view cameras or basic proximity sensors to help manage blind spots. These cameras provide a visual feed to a small display inside the helmet, giving riders a better view of what’s behind them without needing to look over their shoulder. While GPS navigation systems are common for route guidance, their integration into motorcycle safety systems for proactive measures like accident detection or emergency response is limited. Overall, while these traditional systems improve comfort and provide basic enhancements to safety, they lack comprehensive real-time monitoring, proactive accident prevention, and dynamic response capabilities that could significantly enhance rider safety.

3.2.1 DISADVANTAGES

Traditional Helmet Systems: The absence of advanced sensors and communication modules restricts the system's ability to provide a comprehensive safety net. It fails to address emerging road safety challenges, such as alcohol consumption detection, distraction prevention, and real-time accident reporting.
Basic Alcohol Detection Helmets: These helmets, focusing solely on alcohol detection, overlook the broader spectrum of safety concerns. They lack features like distraction monitoring and obstacle avoidance, diminishing their effectiveness in providing holistic road safety.
Conventional Drowsiness Detection Helmets: Reliability issues may arise with basic drowsiness detection methods, leading to false positives or negatives. The system's incapacity to identify nuanced signs of driver fatigue diminishes its overall effectiveness in preventing accidents.
Alarming Helmets for Distraction: inability to differentiate between various types of distractions can result in frequent false alarms. This may lead to rider desensitization to alerts, reducing the system's effectiveness in addressing genuine safety concerns.
Basic Helmet Communication Systems: Limited safety features beyond communication capabilities render these helmets insufficient in addressing diverse safety challenges. Critical aspects like alcohol detection, distraction prevention, and real-time accident reporting are neglected.
GPS-Enabled Helmets: While focused on navigation, these helmets lack a holistic safety approach. Essential features like alcohol detection, distraction prevention, and advanced accident reporting are absent, limiting their effectiveness in enhancing overall road safety.
Limited Obstacle Detection Helmets: The limited functionality of merely alerting the rider to obstacles falls short of a comprehensive safety solution. These helmets lack the capability for automated vehicle control or a well-rounded approach to obstacle avoidance.
Simplified Vibration Sensor Helmets: Relying solely on vibration sensors for accident detection may result in false alarms or overlook subtle impacts. This diminishes the accuracy and reliability of the safety system, potentially undermining its effectiveness in providing timely alerts for genuine accidents.

3.3 PROPOSED SYSTEM:

The proposed smart helmet system is meticulously designed to enhance safety for motorbike riders through a combination of advanced technological features aimed at proactive monitoring and rapid emergency response. At the core of this system is a push button integrated inside the helmet, which the rider must engage to confirm that the helmet is properly seated on their head. This crucial feature ensures that all of the helmet's sophisticated electronics are activated only when the helmet is correctly worn, thereby safeguarding the functionality of the entire system from the outset. Building on this foundation, the helmet employs LeNet Convolutional Neural Network (CNN), implemented using Python, to analyze real-time video footage captured from a small camera and is specifically trained to detect subtle signs of drowsiness, such as frequent blinking, yawning, and other facial cues indicative of fatigue. By identifying these early signs of drowsiness, the system can alert the rider with auditory or visual warnings, prompting them to take necessary precautions or breaks to avoid potential accidents caused by fatigue.

Furthermore, the helmet is equipped with ultrasonic sensors that provide 180-degree monitoring behind the vehicle. These sensors are particularly focused on detecting vehicles in the rider's blind spots, a common cause of accidents on busy roads. By alerting the rider to these hidden dangers, the system greatly enhances situational awareness and aids in preventing collisions. In the event of an accident, the helmet's integrated GPS and GSM modules play a critical role. These modules are triggered automatically when significant impacts are detected by the helmet's sensors, enabling them to send precise geographic location data directly to emergency services and predefined emergency contacts. This ensures that help is dispatched promptly and accurately to the accident scene, potentially reducing the severity of injuries or even saving lives.

Overall, this smart helmet system represents a comprehensive approach to motorbike safety, combining essential features like helmet wear verification, fatigue monitoring, environmental awareness, and emergency response. Each element works synergistically to provide a robust safety net for riders, making motorcycling not only safer but also more responsive to the dynamic conditions of modern roadways. By leveraging cutting-edge technologies such as CNN for drowsiness detection and ultrasonic sensors for obstacle detection, along with reliable communication technologies like GPS and GSM for emergency alerts, the system sets a new standard in protective gear for motorbike riders.

3.3.1 ADVANTAGES:

- Enhanced Safety Through Proactive Monitoring: By incorporating the LeNet Convolutional Neural Network (CNN) to monitor the rider's facial features in real-time, the system can detect early signs of drowsiness or fatigue. This proactive approach allows riders to be alerted before their condition compromises their ability to safely control the motorcycle, thereby preventing potential accidents caused by fatigue.

- Improved Situational Awareness: The integration of ultrasonic sensors around the vehicle provides comprehensive monitoring of the rider's immediate environment, particularly the blind spots. This system enhances the rider's situational awareness by alerting them to nearby vehicles or obstacles that are otherwise difficult to detect, significantly reducing the risk of collisions.

- Rapid Emergency Response: In the event of an accident, the helmet's built-in GPS and GSM modules automatically activate to send the rider’s precise location to emergency services and predefined contacts. This feature ensures that help is dispatched quickly and accurately, which is critical in emergency situations where timely medical intervention can greatly influence the outcome.

- Encouragement of Safe Riding Practices: The requirement for riders to engage a push button to confirm that the helmet is properly worn ensures that the helmet's safety features are active only when it is securely fitted. This system promotes responsible riding habits, as it encourages riders to always wear their helmets correctly for their protection.

- Integration of Advanced Technologies: Utilizing advanced technologies like machine learning (CNN), GPS, and ultrasonic sensors in a motorcycle helmet is an innovative approach that sets a new standard in rider safety equipment. This integration not only boosts the helmet's functional capabilities but also positions it as a cutting-edge solution in the evolving market of motorcycle safety gear.

- User Convenience and Comfort: Despite the complexity of its functionalities, the smart helmet is designed to be user-friendly, with automatic alerts and minimal required input from the rider. This ease of use ensures that riders are not overwhelmed by the technology and can focus fully on the road, enhancing both their comfort and safety.

3.4 ALGORITHMS/METHODOLOGIES USED:

LeNet architecture:The LeNet architecture is a pioneering convolutional neural network (CNN) architecture. It played a crucial role in the advancement of deep learning and laid the foundation for modern CNNs and their applications in image recognition and computer vision tasks.

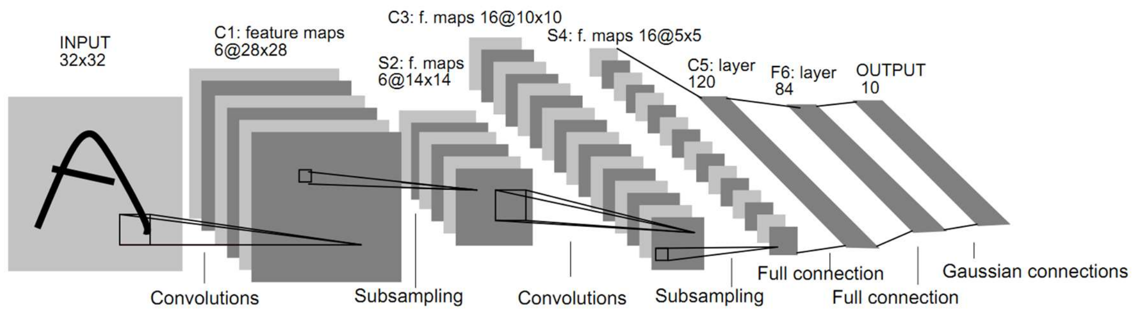


Figure.3.1. LeNet Architecture

WORKING PRINCIPLE:
Input Layer:
Accepts grayscale images of size 32x32 pixels as input.
Convolutional Layer (C1):
Applies six 5x5 filters to the input, generating six feature maps.
Utilizes a sigmoid or hyperbolic tangent activation function.
Pooling Layer (S2):
Performs 2x2 average pooling on each of the six feature maps from C1.
Reduces spatial dimensions.
Convolutional Layer (C3):
Introduces 16 additional 5x5 filters, connected to the subsampled output of S2.
Uses a sigmoid or hyperbolic tangent activation function.
Pooling Layer (S4):
Performs 2x2 average pooling on each of the 16 feature maps from C3.
Fully Connected Layer (C5):
Flattens the output of S4 and connects to a fully connected layer with 120 neurons.
Applies a sigmoid or hyperbolic tangent activation function.
Output Layer:
The final layer, typically consisting of 10 neurons for each digit (0-9).
Applies a softmax activation function for multiclass classification.

The convolution operation convolutes pairs of functions. Here I'm using the plain meaning of "convoluted": to intertwine or twist things together. In the neural network context, the functions we convolute together are the input function P and the kernel function K (remember that kernel is another way to call the receptive field of a feature map). For the 2-dimensional inputs as in LeNet-5, the Pi,j function contains the 2-dimensional values for the input image, which in our case are grayscale values between 0 (white) and 255 (black). The Km,n function contains the 2-dimensional values for the kernel, this is the matrix of weights Wm,n to be learned by the network. The output of the convolution is the feature map Fm,n in the next layer. In practice, the convolution operation is a linear operation, i.e., a weighted sum.

The convolution formula has different forms depending on the context. In the neural network context, we will compute a discrete convolution. The convolution operator is conventionally represented by the $*$ symbol. Hence, we define the convolution between P and K as:

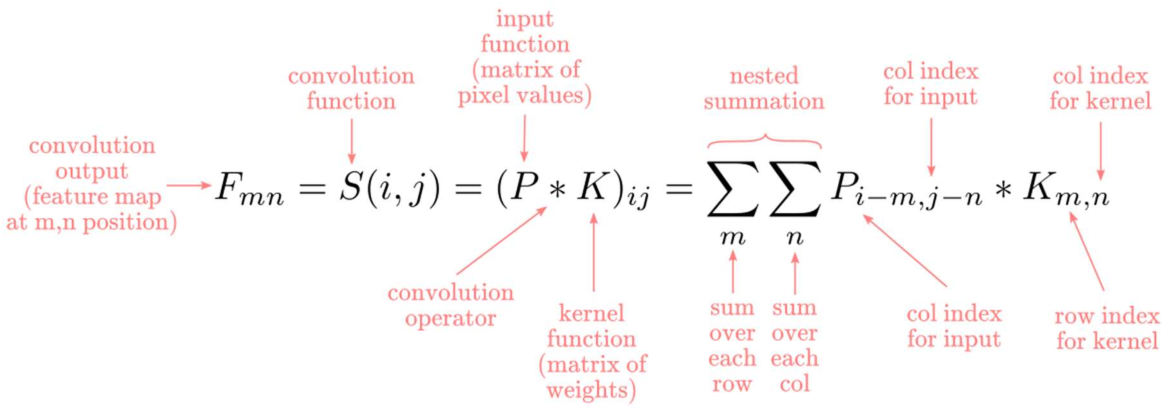
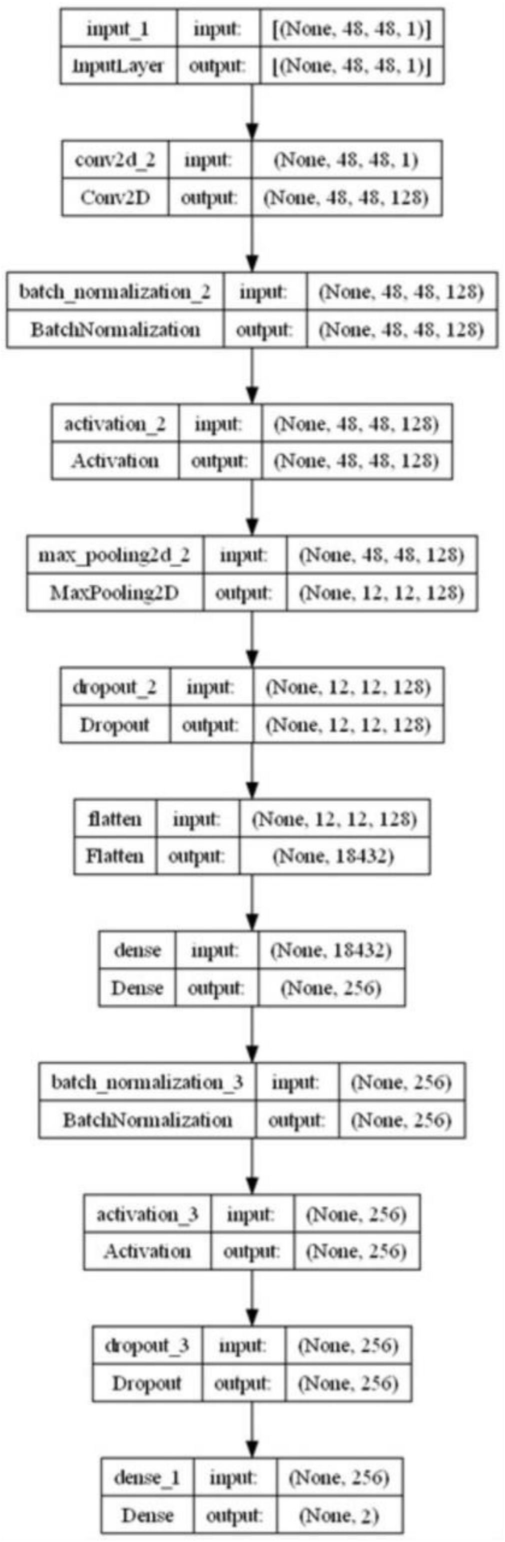

input function (matrix of pixel values)
convolution function
nested summation
col index for input
col index for kernel
convolution output (feature map at m,n position)
$F_{mn} = S(i,j) = (P * K)_{ij} = \sum_m \sum_n P_{i-m,j-n} * K_{m,n}$
convolution operator
kernel function (matrix of weights)
sum over each row
sum over each col
col index for input
row index for kernel

Figure.3.2. Convolution formula

Where i,j are the width and length of the input image, and m,n are the width and length of the kernel.

Using the LeNet convolutional neural network (CNN) architecture for drowsiness detection involves several steps, starting with data collection. A comprehensive dataset of images capturing various facial expressions and eye movements indicative of both alertness and drowsiness is essential. These images should cover diverse conditions such as different lighting, facial orientations, and features from various demographics to ensure robust model training. The images are then pre-processed by resizing them to 48x48 pixels, normalizing pixel values to a 0 to 1 range, and labelling them as 'alert' or 'drowsy'. The LeNet-5 architecture, commonly used for such applications, consists of two convolutional layers, each followed by subsampling (pooling) layers, and three fully connected layers that classify the state of the driver. After training, the model's accuracy is evaluated on a separate test set to determine its effectiveness in identifying drowsiness, using metrics like overall accuracy and a confusion matrix to uncover any biases or misclassifications. Once validated, this model is integrated into a smart helmet system where it processes real-time video from a camera inside the helmet, checks each frame for signs of drowsiness, and triggers an alert if necessary. This deployment within a smart helmet allows for continuous, real-time monitoring of the rider's state, significantly enhancing safety by reducing the risk of accidents due to drowsiness.

LeNet CNN Architecture implementation for drowsiness detection:


input_1 | input: | [(None, 48, 48, 1)]
InputLayer | output: | [(None, 48, 48, 1)]
conv2d_2 | input: | (None, 48, 48, 1)
Conv2D | output: | (None, 48, 48, 128)
batch_normalization_2 | input: | (None, 48, 48, 128)
BatchNormalization | output: | (None, 48, 48, 128)
activation_2 | input: | (None, 48, 48, 128)
Activation | output: | (None, 48, 48, 128)
max_pooling2d_2 | input: | (None, 48, 48, 128)
MaxPooling2D | output: | (None, 12, 12, 128)
dropout_2 | input: | (None, 12, 12, 128)
Dropout | output: | (None, 12, 12, 128)
flatten | input: | (None, 12, 12, 128)
Flatten | output: | (None, 18432)
dense | input: | (None, 18432)
Dense | output: | (None, 256)
batch_normalization_3 | input: | (None, 256)
BatchNormalization | output: | (None, 256)
activation_3 | input: | (None, 256)
Activation | output: | (None, 256)
dropout_3 | input: | (None, 256)
Dropout | output: | (None, 256)
dense_1 | input: | (None, 256)
Dense | output: | (None, 2)
Figure.3.3. Lenet CNN Architecture for drowsiness detection

CHAPTER 4
DESIGN PROCESS

4.1 ARCHITECTURE DIAGRAM:

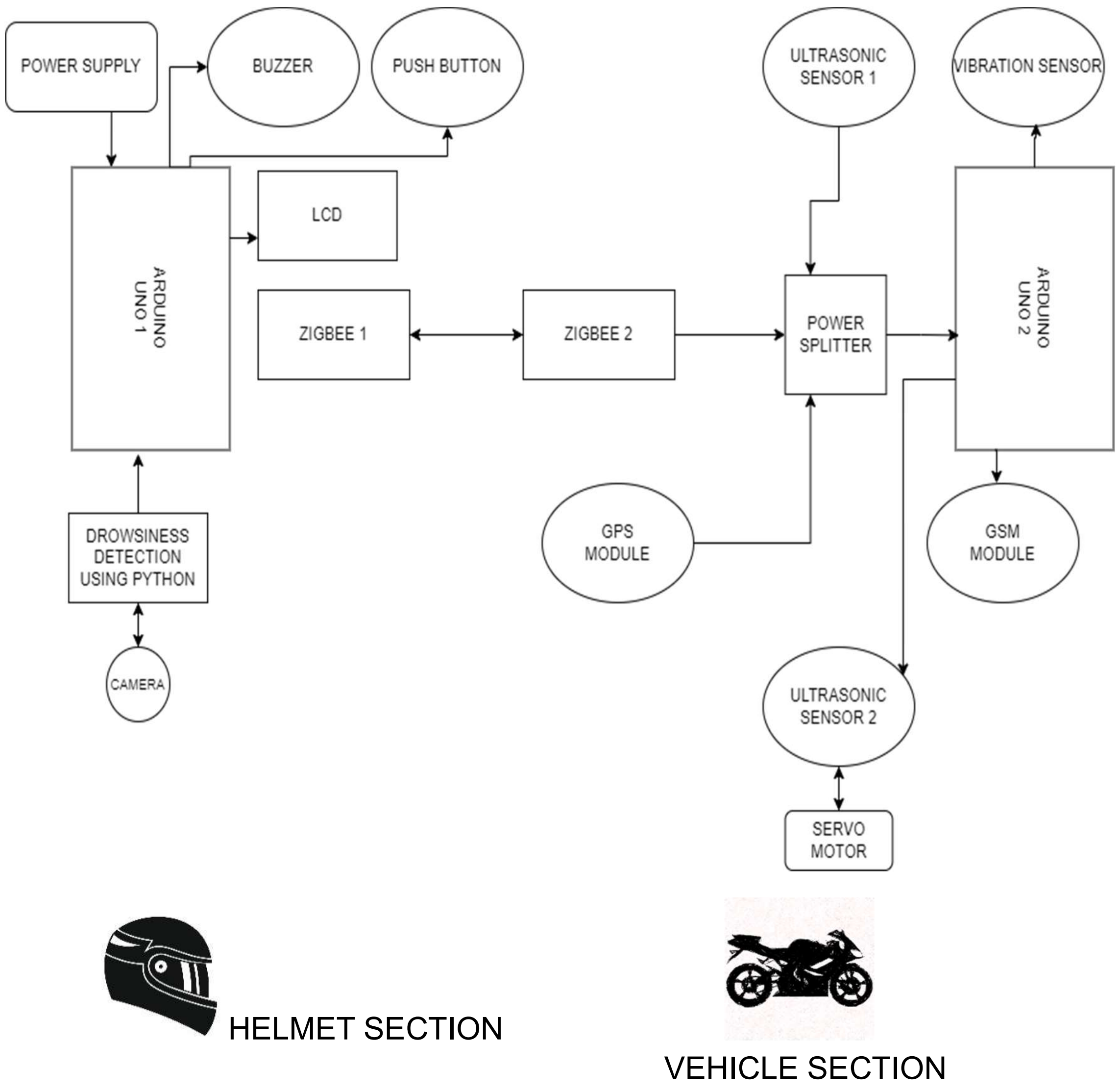


Figure.4.1. Architecture Diagram of the system

As shown in fig.4.1 , there are two sections of the system comprising of the Helmet section and the vehicle section. The helmet section is designed around an Arduino microcontroller, which serves as the central processing unit, orchestrating the functionalities of various interconnected components. Essential to the system is a reliable power supply that powers the Arduino along with all peripheral devices. Among these peripherals, a buzzer is integrated to provide audible alerts to the rider, crucial for warnings such as drowsiness detection or imminent collision threats.

An LCD display enhances user interaction by visually conveying important information and alerts, such as system status and safety notifications, directly to the rider. For robust inter-device communication, a Zigbee communication module is included to facilitate seamless wireless connectivity between the helmet and the vehicle’s onboard systems.

This setup allows for continuous exchange of data and alerts, ensuring that safety features are dynamically updated based on real-time environmental and vehicular sensor inputs. Together, these components form a cohesive unit where the Arduino continuously monitors input from both internal and external sensors, processes this data to detect any conditions warranting immediate attention, and activates appropriate alerts via the buzzer and LCD display. Additionally, the Zigbee module plays a crucial role in maintaining synchronized safety monitoring and alert systems between the helmet and the vehicle, enhancing the overall effectiveness of the rider’s safety gear.

The vehicle section of the smart helmet system is equipped with several sensor components that work together to enhance vehicle safety and response capabilities. Central to these features are ultrasonic sensors, which are strategically mounted around the vehicle to monitor blind spots.

These sensors are crucial for detecting objects or other vehicles that may not be visible through standard rear-view mirrors, thereby significantly reducing the likelihood of side collisions by alerting the rider to potential hazards.To complement the ultrasonic sensors, a servo motor is integrated to enable a 180-degree range of motion. This allows the sensors to sweep across a broad area, providing comprehensive coverage around the vehicle. This setup ensures that the rider has a fuller understanding of the surrounding environment, increasing safety when maneuvering or changing lanes.

In the event of an accident or emergency, the vehicle section is also equipped with GPS and GSM modules. The GPS module is responsible for acquiring precise location data, which is critical during emergencies for pinpointing the vehicle’s exact position. The GSM module uses this location data to send alerts to predefined emergency contacts or services. This automated message system is activated under specific conditions, such as when a significant impact is detected by the vehicle’s sensors, ensuring that help can be dispatched quickly to the accident site.Furthermore a power splitter is used to generate power to all sensor components and the vehicle incorporates a motor flux controller that regulates the vehicle's movement especially under dynamic riding conditions or in response to alerts generated by the safety system.

4.2 DATA FLOW DIAGRAM:

4.2.1. Flow diagram of drowsiness detection system

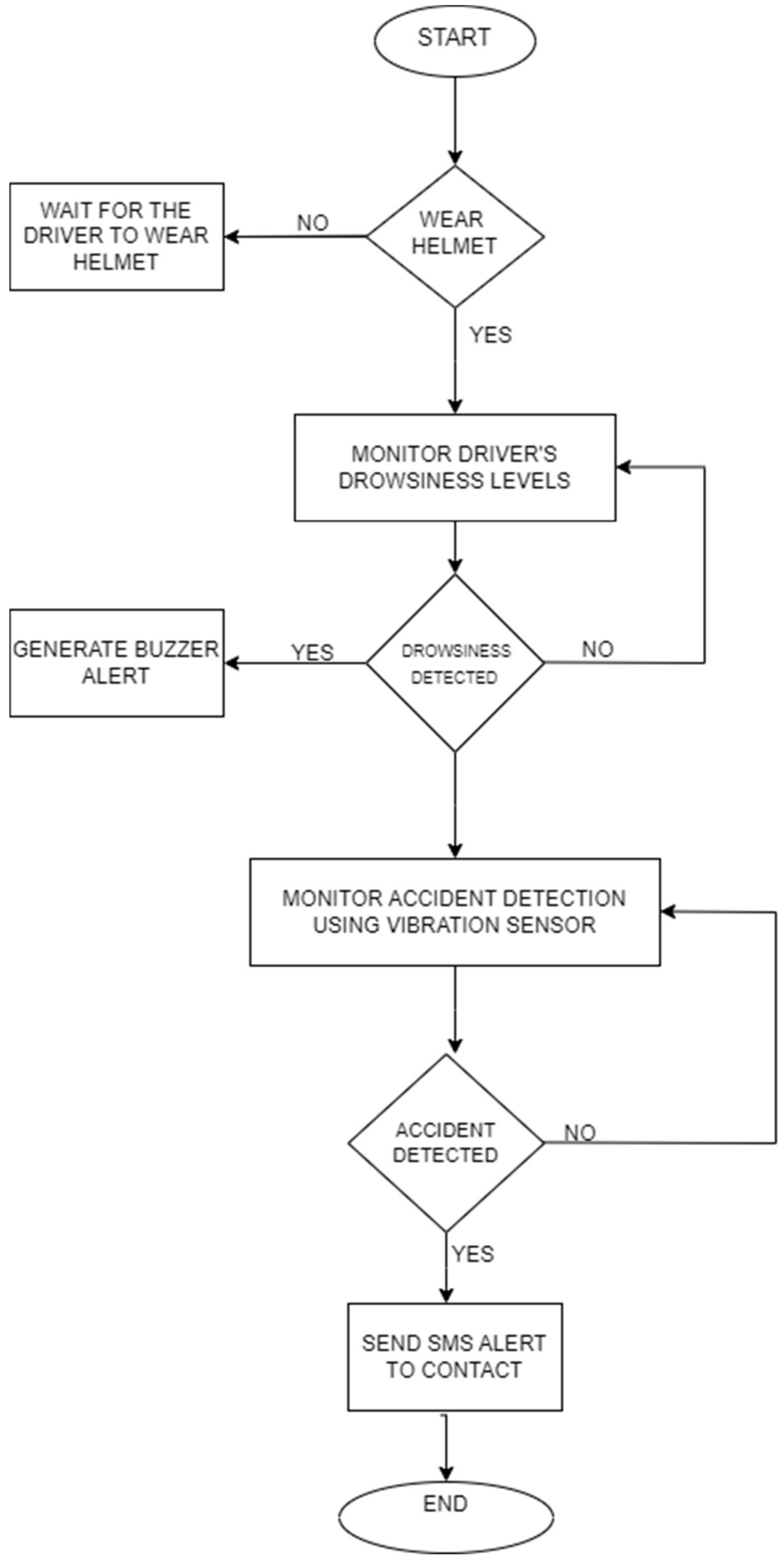

START
WEAR HELMET
NO
WAIT FOR THE DRIVER TO WEAR HELMET
YES
MONITOR DRIVER'S DROWSINESS LEVELS
DROWSINESS DETECTED
YES
GENERATE BUZZER ALERT
NO
MONITOR ACCIDENT DETECTION USING VIBRATION SENSOR
ACCIDENT DETECTED
NO
YES
SEND SMS ALERT TO CONTACT
END

Figure.4.2. Flow diagram of drowsiness detection system

4.2.2 Flow diagram of blind spot detection system:

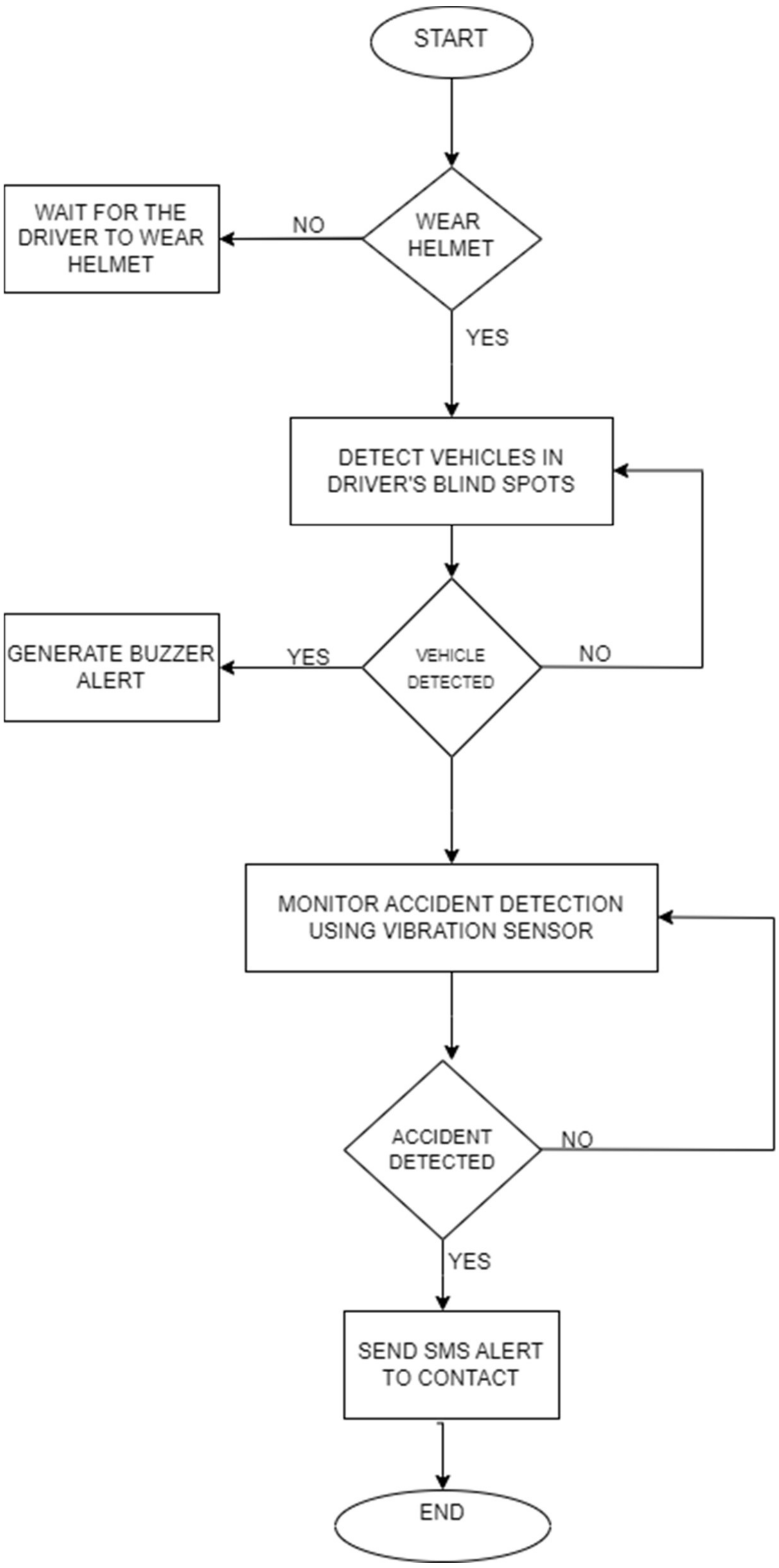

START
WEAR HELMET
NO
WAIT FOR THE DRIVER TO WEAR HELMET
YES
DETECT VEHICLES IN DRIVER'S BLIND SPOTS
VEHICLE DETECTED
YES
GENERATE BUZZER ALERT
NO
MONITOR ACCIDENT DETECTION USING VIBRATION SENSOR
ACCIDENT DETECTED
NO
YES
SEND SMS ALERT TO CONTACT
END

Figure.4.3 Flow diagram of blind spot detection system

4.3 SYSTEM REQUIREMENTS

4.3.1 SOFTWARE REQUIREMENTS:

The specifications of the software components used in proposed systems are shown in the table

Table 4.1 Software requirements

SOFTWARE	SPECIFICATION
Operating system	Windows 7 or higher
Language	Embedded C,python,
Packages	Tensorflow, imutils
IDE	VS Code, Arduino IDE, Jupyter notebook

4.3.2 HARDWARE REQUIREMENTS:

The specifications of the hardware components used in proposed system are shown in the table

Table 4.2 Hardware requirements

Arduino (2)	DC motor
Zigbee(2)	Servo motor
HC-SR04 ultrasonic sensor	Camera
LCD	Push button
Vibration sensor	Power splitter
GPS Module	Power supply
GSM module	Buzzer
Helmet	Vehicle board

4.4 SOFTWARE DESCRIPTION:

4.4.1 EMBEDDED C:

Embedded C is most popular programming language in software field for developing electronic gadgets. Each processor used in electronic system is associated with embedded software. Embedded C programming plays a key role in performing specific function by the processor. In day-to-day life we used many electronic devices such as mobile phone, washing machine, digital camera, etc. These all device working is based on microcontroller that are programmed by embedded C.
The Embedded C code written in above block diagram is used for blinking the LED connected with Port0 of microcontroller.
EMBEDDED SYSTEMS: Embedded System is a system composed of hardware, application software and real time operating system. It can be small independent system or large combinational system. Our Embedded System tutorial includes all topics of Embedded System such as characteristics, designing, processors, microcontrollers, tools, addressing modes, assembly language, interrupts, embedded c programming, led blinking, serial communication, LCD programming, project implementation etc.
It is an arrangement in which all the unit combined to perform a work together by following certain set of rules in real time computation. It can also be defined as a way of working, organizing or doing one or many tasks according to a fixed plan.
An Embedded System is a system that has software embedded into computer-hardware, which makes a system dedicated for a variety of application or specific part of an application or product or part of a larger system.
An embedded system can be a small independent system or a large combinational system. It is a microcontroller-based control system used to perform a specific task of operation.
An embedded system is a combination of three major components:

- Hardware: Hardware is physically used component that is physically connected with an embedded system. It comprises of microcontroller based integrated circuit, power supply, LCD display etc.
- Application software: Application software allows the user to perform varieties of application to be run on an embedded system by changing the code installed in an embedded system.
- Real Time Operating system (RTOS): RTOS supervises the way an embedded system work. It act as an interface between hardware and application software which supervises the application software and provide mechanism to let the processor run on the basis of scheduling for controlling the effect of latencies.

4.4.2 PYTHON:

Python is an essential tool due to its robustness, flexibility, and the comprehensive ecosystem of libraries designed for data science and machine learning. Python libraries such as TensorFlow and Keras provide a high-level, user-friendly interface for building and training deep learning models. These libraries simplify complex processes like constructing convolutional layers, applying pooling operations, and implementing activation functions, which are integral to the CNN architecture.The process begins with the collection and preprocessing of image data, where Python's tools like NumPy and Matplotlib can be used for manipulating and visualizing the images, ensuring they are properly formatted and normalized for neural network input.
The core of Python's application in this system lies in efficiently training the model on a substantial dataset of labeled images, using backpropagation algorithms provided by TensorFlow to adjust weights and biases according to the error rates. Keras facilitates this with functions that handle the training loops, including feedforward operation and backpropagation, with minimal code. After training, Python's capabilities allow for easy testing and validation of the model's accuracy and performance on unseen data, providing metrics that help evaluate its effectiveness in real-world scenarios.
Moreover, Python supports the deployment phase, where the trained model can be integrated into the helmet's software system to function in real-time. With libraries such as OpenCV for image processing and TensorFlow Lite for deploying machine learning models on mobile and embedded devices, Python ensures that the transition from a training environment to a practical, operational system is smooth.Thus, Python's comprehensive environment not only supports the initial development stages but also extends to ongoing maintenance and real-time application, making it indispensable for building advanced safety features in automotive technologies.

4.4.3 VS Code:

VS Code is used as an Integrated Development Enivronment (IDE) for writing and managing code. It provides features such as code editing, syntax highlighting, auto- completion and debugging. Additionally, possible use extensions and plugins to enhance its functionality, integrate with version control systems like Git, and execute commands within the integrated terminal. Overall VS Code improves coding experience and streamlines for development workflow.

4.4.4 Arduino IDE:

The Arduino Integrated Development Environment (IDE) is a crucial software tool for programming Arduino boards, which are widely used in hobbyist electronics, education, and prototyping. The Arduino IDE allows users to write, compile, and upload code to Arduino microcontroller boards from a straightforward, user-friendly interface. It supports the Arduino programming language, which is based on C/C++, offering built-in functions and libraries that simplify interacting with hardware components like sensors, motors, and displays. Users can quickly start programming by accessing numerous example sketches and libraries that demonstrate basic and advanced functionalities. The IDE includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, which enhance coding efficiency and readability. Additionally, it provides a serial monitor for debugging and displaying data from the Arduino to the computer, aiding in the development and troubleshooting of projects. By bridging the gap between hardware and software, the Arduino IDE enables enthusiasts and professionals alike to develop interactive projects and applications with ease.

For the smart helmet project, the Arduino IDE provides invaluable tools. The code editor within the IDE offers syntax highlighting, auto-indentation, and brace matching, making it easier to write error-free code efficiently. It also features example sketches that help developers quickly understand how to interface with different hardware components. This is particularly useful for controlling the servo motors attached to the ultrasonic sensors, enabling them to cover broader angles for more effective blind spot monitoring.

The IDE also includes a robust library management system, allowing users to include and manage libraries essential for handling GPS data, managing GSM communication, or integrating machine learning algorithms for drowsiness detection.

Furthermore, the serial monitor tool within the Arduino IDE is crucial for debugging and real-time data monitoring. It allows developers to view output from the Arduino, such as sensor readings or diagnostic messages, which are critical during the development and testing phases of the smart helmet. This feature ensures that developers can iteratively test and refine each component of the helmet, from sensor accuracy to communication reliability, thus enhancing the overall system's performance and safety features.

Overall, the Arduino IDE not only supports the basic programming needs but also enhances the development process of technologically advanced projects like the smart helmet by streamlining code development, simplifying hardware integration, and providing real-time debugging tools. This comprehensive suite of features makes it an indispensable tool in the creation and refinement of interactive and safety-oriented devices.

4.4.5 Jupyter Notebook:

Jupyter Notebook is an incredibly versatile and interactive web-based interface widely used for developing machine learning models, including convolutional neural networks (CNNs). It provides a user-friendly environment where developers can write and execute code, visualize data, and see results instantaneously, all within the same document. This platform supports various programming languages, notably Python, which is extensively used in machine learning for its rich ecosystem of libraries such as TensorFlow, Keras, and PyTorch.

When training CNN models for tasks like image recognition or drowsiness detection in smart helmet projects, Jupyter Notebook allows for an iterative approach to development. Developers can write code to load and preprocess data, define and train the CNN, and evaluate its performance, segment by segment. Each code block can be run independently, making it easy to test different parameters or algorithms without rerunning the entire script. This is particularly useful in machine learning, where tuning model parameters is a frequent task.

Additionally, Jupyter Notebook includes visualization libraries like Matplotlib and Seaborn, which are crucial for examining data distributions, visualizing the architecture of models, and plotting performance metrics such as loss and accuracy during training phases. These visual tools help in diagnosing problems early in the model training process, such as overfitting or underfitting.

4.5 HARDWARE DESCRIPTION:

4.5.1 Arduino UNO:

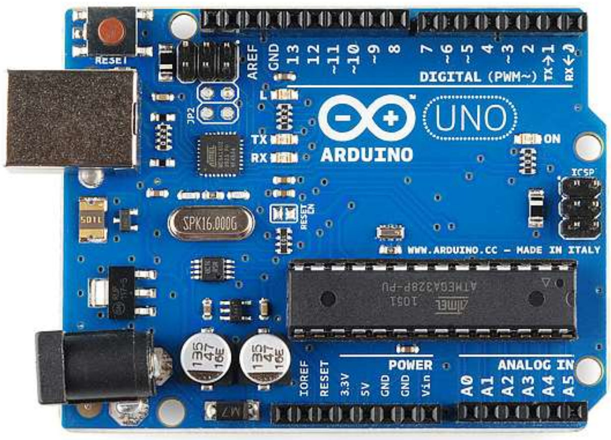


Figure.4.4 Arduino microcontroller

The Arduino microcontroller is a cornerstone in the world of electronics and embedded systems, favored for its ease of use and flexibility. Essentially, it is a small, powerful computer that can be programmed to sense and control objects in the physical world. At its core, the Arduino operates by reading inputs - light on a sensor, a finger on a button, or a Twitter message - and turning it into an output - activating a motor, turning on an LED, publishing something online. Users can tell their Arduino what to do by writing code in the Arduino programming language and using the Arduino development environment. The microcontroller on the board is programmed using the Arduino IDE (Integrated Development Environment) over a standard USB connection. Once programmed, the board can execute the code autonomously without the need for an additional computer. This autonomy combined with the ability to communicate with sensors, actuators, and other devices makes it incredibly useful for countless automation projects.

4.5.2 Zigbee:

ZigBee is a wireless technology developed as an open global standard to address the unique needs of low-cost, low-power, wireless sensor networks. The standard takes full advantage of the IEEE 802.15.4 physical radio specification and operates in unlicensed bands worldwide at the following frequencies: 2.400–2.484 GHz, 902-928 MHz and 868.0–868.6 MHz.

- The power levels (down from 5v to 3.3v) to power the zigbee module.
- The communication lines (TX, RX, DIN and DOUT) to the appropriate voltages.

The Zigbee module acts as both transmitter and receiver. The Rx and Tx pins of ZIGBEE are connected to Tx and Rx of microcontroller respectively. The data's from microcontroller is serially transmitted to Zigbee module via UART port. Then Zigbee transmits the data to another Zigbee. The data's from Zigbee transmitted from Dout pin. The Zigbee from other side receives the data via Din pin.

4.5.3 Power supply:

Powering Arduino projects effectively is crucial for their stability and performance. Charging adapters are commonly used as power supplies for Arduino microcontrollers because they provide a reliable and steady voltage, ensuring the Arduino operates within safe parameters. Typically, these adapters can convert AC power from a wall outlet into a DC voltage suitable for the Arduino, usually 5V or 12V depending on the specific requirements of the project. The use of a charging adapter is particularly beneficial because it can deliver a consistent power supply, which is essential for projects that involve sensitive components like sensors and motors that may malfunction or provide inaccurate readings under fluctuating power conditions.

4.5.4 LCD (Liquid Crystal Display):

A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs. A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data.The command register stores the command instructions given to the LCD.

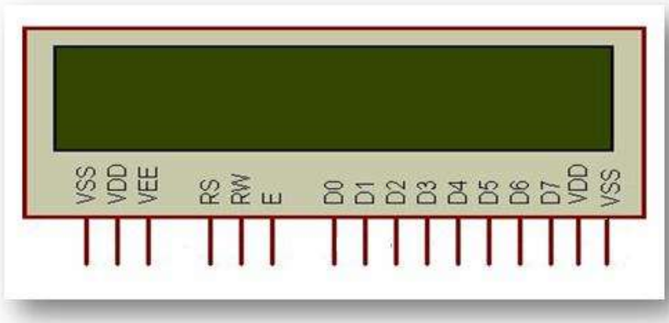


Figure.4.5 LCD

PIN DESCRIPTION:

Table 4.3 LCD PIN DESCRIPTION

Pin No	Function	Name
1	Ground (0V)	Ground
2	Supply voltage; 5V (4.7V – 5.3V)	Vcc
3	Contrast adjustment; through a variable resistor	V_{EE}
4	Selects command register when low; and data register when high	Register Select
5	Low to write to the register; High to read from the register	Read/write
6	Sends data to data pins when a high to low pulse is given	Enable
7	8-bit data pins	DB0
8		DB1
9		DB2

10		DB3
11		DB4
12		DB5
13		DB6
14		DB7
15	Backlight V_{CC} (5V)	Led+
16	Backlight Ground (0V)	Led-

4.5.5 VIBRATION SENSOR:

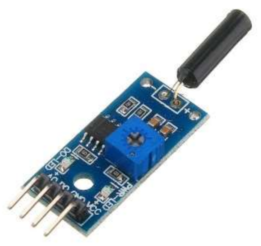


Figure 4.6 VIBRATION SENSOR

Vibration Sensor is a high sensitivity non-directional vibration sensor. When the module is stable, the circuit is turned on and the output is high. When the movement or vibration occurs, the circuit will be briefly disconnected and output low. At the same time, you can also adjust the sensitivity according to your own needs. The vibration switch that opens when vibration is detected and closes when there is no vibration.

4.5.6 HC-SR04 ULTRASONIC SENSOR:

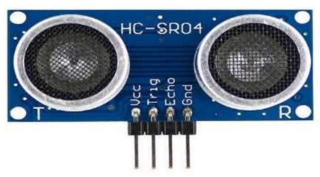


Figure 4.7 HC-SR04 ULTRASONIC SENSOR

The HC-SR04 ultrasonic sensor uses sonar to determine distance to an object like bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package. It comes complete with ultrasonic transmitter and receiver modules.as the name indicates measure distance by using ultrasonic waves .the sensor head emits ultrasonic wave and receives the wave reflected back from the target .Ultrasonic measures the distance through the target by measuring time between emission and reception.

4.5.7 GPS AND GSM MODULES:

Figure.4.8 GPS

Figure 4.9 GSM

GPS (Global Positioning System) and GSM (Global System for Mobile Communications) modules play a pivotal role in enhancing the capabilities of various technology applications, particularly in the realm of communications and location-based services. The GPS module provides precise geographical positioning information by receiving signals from a constellation of satellites orbiting the Earth. This feature is indispensable for applications that require real-time location tracking, such as navigation systems, geocaching, fleet management, and emergency services. By offering exact coordinates, speed, and time data, GPS enables devices to perform complex tasks such as route planning, asset tracking, and geographic data mapping with high accuracy.

On the other hand, GSM modules facilitate mobile communication using cellular technology, allowing devices to send and receive messages, make calls, and transfer data over a mobile network. This functionality is crucial for IoT (Internet of Things) projects, enabling remote monitoring and control of devices spread over wide areas. In emergency response systems, for instance, a GSM module can automatically send alerts or communicate distress signals to emergency services, providing vital information like location coordinates derived from the GPS module. The integration of GPS and GSM in technology projects thus offers a robust solution for creating systems that not only understand their position in the world but can also communicate it to other systems and users effectively, enhancing safety, efficiency, and connectivity.

4.5.8 BUZZER
Buzzers are compact audio signaling devices commonly used in various electronic projects and products to provide audible alerts, notifications, or feedback to users. They are especially useful in applications where visual communication might be insufficient or impractical. In many devices, including household appliances, computer systems, and alarm systems, buzzers serve to alert users about specific conditions or required actions. For instance, they

may signal the completion of a washing cycle, warn about a system error, or alert in response to security breaches.

4.5.9 SERVO MOTOR:

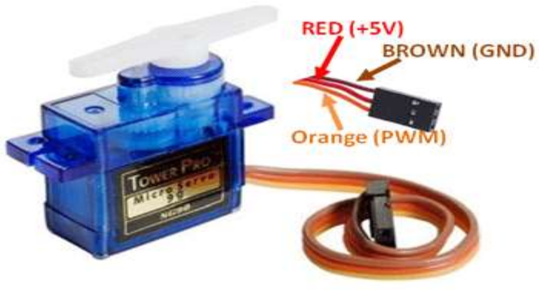


Figure 4.10 SERVO MOTOR

A Servo Motor is a small device that has an output shaft. This shaft can be positioned to specific angular positions by sending the servo a coded signal. As long as the coded signal exists on the input line, the servo will maintain the angular position of the shaft. If the coded signal changes, the angular position of the shaft changes. In practice, servos are used in radio-controlled airplanes to position control surfaces like the elevators and rudders. They are also used in radio-controlled cars, puppets, and of course, robots.
Servo motors use feedback to determine the position of the shaft, you can control that position very precisely. As a result, servo motors are used to control the position of objects, rotate objects, move legs, arms or hands of robots, move sensors etc. with high precision. Servo motors are small in size, and because they have built-in circuitry to control their movement, they can be connected directly to an Arduino. Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller..

4.5.10 PUSH BUTTON AND CAMERA :

A push button can serve multiple purposes, such as activating the system, confirming settings, or even sending a distress signal manually. The simplicity of a push button makes it highly effective for scenarios where quick and uncomplicated user interaction is necessary. Cameras, on the other hand, offer sophisticated visual input capabilities that are essential for features like drowsiness detection or obstacle recognition in smart helmets. In such applications, a camera continuously captures video footage of the rider's face to monitor for signs of fatigue using image processing algorithms. This allows the system to analyze facial expressions and eye movements to detect early signs of drowsiness, triggering alerts to ensure the rider's safety.

4.6 MODULE DESCRIPTION:

The various modules present in the proposed system are:

1. Drowsiness Detection using LeNet CNN
2. Helmet Designing
3. Blind spot detection in vehicle section
4. GPS and GSM modules

4.6.1 Drowsiness Detection using LeNet CNN:

The first module of the smart helmet system focuses on drowsiness detection, utilizing the LeNet Convolutional Neural Network (CNN) architecture. This module is essential for enhancing rider safety by actively monitoring signs of fatigue that could lead to accidents if unchecked. Here's a detailed explanation of how this module functions:

Components:

- Embedded Camera: A small camera is integrated within the helmet, positioned to capture the rider's face continuously. This camera feeds live video data, focusing particularly on the rider's eyes and facial expressions, which are critical indicators of drowsiness.
- Processing Unit: This includes a microprocessor or microcontroller capable of running the CNN model. It processes the video input from the camera.

Functionality:

- Image Capture: The camera captures sequential images or video frames of the rider's face during the ride. These images are vital for analyzing changes in facial expressions associated with fatigue, such as drooping eyelids, frequent blinking, and yawning.
- Image Preprocessing: Before feeding them into the CNN, the images are preprocessed to adjust size, contrast, and brightness to ensure uniformity and improve the model's accuracy. This preprocessing might include normalizing the image pixels to a scale that the model can efficiently process.

LeNet CNN Architecture: The LeNet architecture is particularly suited for image recognition tasks due to its relatively simple layout, making it ideal for real-time applications on systems with limited processing capabilities. The model typically consists of two convolutional layers, each followed by average pooling layers, and fully connected layers at the end. The convolutional layers help in extracting various features from the images, such as edges, textures, and specific shapes associated with drowsiness.

Drowsiness Detection: The CNN processes the preprocessed images to detect patterns indicative of drowsiness. It uses the learned weights from training on a dataset of labeled images (images tagged as 'drowsy' or 'alert') to predict the current state of the rider.

Alert System: If the CNN predicts a 'drowsy' state, the system triggers an alert mechanism. This could involve activating a buzzer within the helmet and displaying a warning message on an internal screen, urging the rider to take a break or alerting them to the potential danger.

Importance:

This drowsiness detection module plays a crucial role in preemptively identifying signs of fatigue, providing real-time interventions that can significantly reduce the risk of accidents caused by drowsiness. By leveraging the LeNet CNN, which is efficient in processing and accurate in prediction, the system ensures that these critical alerts are both timely and reliable, directly contributing to the rider's safety and well-being during travel.

4.6.2 Helmet Designing:

Designing the Helmet with Arduino, Buzzer, and LCD

Components:

- Arduino Microcontroller: Acts as the central processing unit within the helmet, managing inputs from sensors and controlling outputs like the buzzer and LCD.
- Buzzer: Used for audible alerts to notify the rider about potential dangers or system warnings.
- LCD Display: A small screen integrated into the helmet to display visual information such as navigation data, system status, or safety warnings.

Design Considerations:

- Integration: The Arduino microcontroller is programmed to process input data from various sensors and execute response actions. For instance, if the drowsiness detection system triggers an alert, the Arduino will activate the buzzer and display a warning message on the LCD.
- Output Management: The Arduino controls when and how alerts are issued. It decides based on sensor inputs and Zigbee communications whether to sound the buzzer and what information to display on the LCD.
- Power Supply: Ensuring that the helmet has a reliable power source, likely a rechargeable battery, is crucial. It must be capable of powering the Arduino, sensors, communication modules, buzzer, and LCD throughout typical usage periods.

Importance:

This setup ensures that the helmet not only serves as a protective gear but also as an interactive device that actively contributes to the rider’s safety through advanced technology. The integration of Arduino allows for flexibility in programming and expansion, the buzzer provides immediate auditory feedback, and the LCD offers a direct visual interface with the rider, enhancing the overall functionality of the helmet system.

To enhance the smart helmet system with seamless and reliable communication between the helmet and the vehicle, a dedicated Zigbee Communication Module can be integrated. This module will facilitate efficient data transfer and control commands, ensuring that all safety features are synchronized between the rider's helmet and the vehicle. Here’s a detailed breakdown of this module:

Zigbee Communication Module

Components:

- Zigbee Transceiver: Both the helmet and the vehicle are equipped with Zigbee transceivers. These devices are capable of sending and receiving data over a secure, low-power wireless network, making them ideal for IoT applications where energy efficiency and reliable communication are crucial.
- Microcontroller Interface: The transceivers are connected to the respective microcontrollers in both the helmet and the vehicle, facilitating the integration of Zigbee into the system's overall electronic architecture.

Functionality:

- Real-Time Data Exchange: The Zigbee module ensures continuous communication between the helmet and the vehicle. This includes sending alerts from the helmet to the vehicle (such as drowsiness or collision warnings) and vehicle status information to the helmet (like blind spot detections or vehicle diagnostics).
- Network Reliability and Range: Zigbee is known for its ability to form mesh networks, which enhances the communication range and reliability. This is particularly useful in scenarios where direct communication might be obstructed or when the vehicle and rider are at different orientations.

Importance:

The Zigbee Communication Module plays a vital role in maintaining constant and reliable communication links within the smart helmet ecosystem. This connectivity ensures that all components work harmoniously, providing timely alerts and updates that significantly enhance rider safety.

4.6.3 Blind spot detection in vehicle section:

The next module to consider in the smart helmet system design is the Collision Avoidance Module, which significantly enhances safety by helping riders detect and avoid potential hazards in their immediate environment. Here’s an overview of this module:

Components:

- Ultrasonic Sensors: These are installed around the helmet (or on the vehicle depending on design preference) to detect objects and other vehicles in close proximity, particularly in the blind spots.
- Microcontroller: Manages data from the ultrasonic sensors and processes the information to determine the presence of potential hazards.

Functionality:

- Distance Measurement: Ultrasonic sensors emit sound waves that bounce back when they hit an object. By calculating the time it takes for the echo to return, the sensors determine the distance to nearby objects.
- Hazard Detection: The microcontroller analyzes the distance data to identify potential collisions. If an object is within a critical distance, it is flagged as a hazard.
- Alert System: Once a hazard is detected, the system triggers an alert mechanism. This can include auditory warnings (such as beeps) and visual signals (LEDs on the helmet or alerts on an internal display), informing the rider of the danger and allowing them to react accordingly.

Importance:

The Collision Avoidance Module is crucial for providing real-time situational awareness to riders, especially in complex environments like urban traffic where blind spots and sudden obstacles can lead to accidents. By integrating this module, the smart helmet not only enhances the rider's ability to perceive and react to immediate physical threats but also significantly boosts overall traffic safety. This proactive approach to collision avoidance is particularly valuable in enhancing the defensive driving capabilities of motorcyclists.

4.6.4 GPS and GSM modules:

For a comprehensive smart helmet system, the integration of GPS and GSM modules in the vehicle section plays a crucial role in ensuring robust safety and communication functionalities. Here’s a detailed explanation of these modules:

GPS Module for Real-Time Location Tracking
Components: The module includes a GPS receiver installed within the vehicle.

Functionality:

- Location Tracking: The GPS receiver continually acquires satellite signals to determine the precise geographical location of the vehicle. This information is crucial for navigation and in the event of an emergency.
- Data Integration: Location data can be integrated with the helmet’s display system to provide real-time navigation assistance directly to the rider, enhancing travel experience and safety.

Importance:
This module ensures that the vehicle's exact location is always available, which is vital not only for navigation purposes but also for safety measures, such as providing location updates to emergency responders if an accident occurs.

4.6.5 GSM Module for Communication:

Components: Incorporates a GSM modem connected to a microcontroller within the vehicle, with a SIM card for cellular communication.
Functionality:

- Emergency Alerts: In the event of an accident detected by the vehicle's sensors or helmet’s alert system, the GSM module can automatically send text messages or make calls to pre-configured emergency contacts or services. This alert includes the precise location provided by the GPS module.
- Data Transmission: Besides emergency uses, the GSM module can transmit various telematics data such as travel speed, trip logs, and other performance metrics to a secured server or directly to the rider's smartphone app for further analysis or for keeping a riding diary.

Importance:
The GSM module provides a critical communication link between the vehicle and external services or devices. It ensures that in the case of an emergency, help can be contacted automatically and swiftly, greatly increasing the rider’s safety. Additionally, it supports data services that can enhance the user experience by allowing riders to track their trips and analyze their riding patterns.

Together, the GPS and GSM modules form a powerful duo in the vehicle section of the smart helmet system, enhancing both navigational capabilities and safety features. These modules ensure that the rider is well-informed, well-connected, and safer during travels, making them indispensable components of a modern smart helmet system.

CHAPTER 5
IMPLEMENTATION

5.1 Planning and Component Acquisition

5.1.1 Identify Requirements and Specifications:

The initial step involves conducting a thorough analysis to outline the technical and user requirements of the smart helmet system. This includes detailed discussions with potential users, safety experts, and stakeholders to identify crucial features and functionalities that the helmet must possess. Particular attention is paid to regulatory requirements and industry standards for motorcycle safety equipment to ensure compliance throughout the design and development process.

- Safety Requirements: Specifications such as impact resistance, weatherproofing, and ergonomic design are defined to ensure the helmet not only provides advanced technological features but also meets traditional safety standards.
- Technical Specifications: Parameters for the electronic components, such as processing power for the Arduino, the range and sensitivity for the ultrasonic sensors, image resolution and frame rate for the camera, and power requirements for the overall system, are established. This phase also defines the communication protocols between the helmet and vehicle, ensuring compatibility and robust data exchange capabilities.
- User Experience Needs: User interface (UI) requirements, ease of use, aesthetics, and comfort are considered. Features such as the readability of the LCD display under various lighting conditions, the audibility of the buzzer in noisy environments, and the user-friendliness of any manual controls on the helmet are outlined.

5.1.2 Component Sourcing and Procurement:

With the specifications in hand, the next step is sourcing the necessary components that meet the defined criteria. Each component's selection is critical as it affects the overall functionality, reliability, and cost-effectiveness of the smart helmet.

  - Vendor Selection: Partners and suppliers are carefully selected based on their ability to provide high-quality components that meet the helmet's technical specifications. This includes evaluating suppliers for the Arduino board, Zigbee modules, sensors, and other electronic parts. Vendor reliability, terms of service, warranty provisions, and after-sales support are also considered in the selection process.

- Component Testing: Before final procurement, sample components from selected vendors are tested to verify their performance. This testing might include bench tests for electronic components, field tests for sensors, and lab tests for material durability.
- Bulk Procurement: Once the suppliers are finalized and sample components meet the testing standards, bulk orders are placed. This stage also involves negotiating prices and delivery schedules to align with the project timeline and budget. It is crucial to establish agreements that allow for flexibility in order modifications and returns in case subsequent tests during integration reveal issues.
- Inventory Management: As components arrive, they are cataloged and stored properly to prevent damage and ease accessibility during the development phase. An inventory management system is set up to track component usage, maintain reorder levels, and manage logistics from the storage area to the development lab.

5.2 Module Development and Testing

Drowsiness Detection Module:

The development of the drowsiness detection module entails several iterative steps. Firstly, a suitable machine learning framework is chosen for implementing the LeNet CNN architecture. Data preprocessing techniques are applied to enhance model performance, such as image normalization and augmentation. The model is trained using a diverse dataset of annotated images, iteratively fine-tuning hyperparameters to optimize accuracy and generalization. Rigorous testing is conducted to evaluate the model's robustness under various lighting conditions, facial expressions, and head movements.

Helmet Designing:

The helmet's design involves ergonomic considerations, ensuring comfort, fit, and aerodynamics. The integration of electronics requires meticulous wiring and soldering to maintain electrical integrity and minimize interference. Extensive user testing is conducted to assess comfort, usability, and safety before finalizing the design.

Blind Spot Detection in Vehicle Section:

Integrating ultrasonic sensors for blind spot detection involves mounting sensors strategically on the vehicle chassis to maximize coverage while minimizing blind spots. Calibration procedures are performed to ensure accurate distance measurements and reliable obstacle detection. Signal processing algorithms are implemented to filter sensor data and identify potential hazards based on proximity and relative velocity. Real-world testing in diverse driving conditions is essential to validate the system's performance and optimize parameters for optimal sensitivity and specificity.

GPS and GSM Modules

The integration of GPS and GSM modules requires careful consideration of antenna placement and signal interference. Signal strength and reception quality are assessed through field testing to identify optimal mounting locations. Software development focuses on implementing communication protocols for data exchange and error handling mechanisms to ensure data integrity. Integration with external APIs may be necessary to access additional functionalities such as geolocation services or emergency contact databases. Extensive testing is conducted to validate location accuracy, transmission reliability, and battery efficiency under various operating conditions.

5.3 System Integration and Pilot Testing

Integration

Integrating individual modules into a cohesive system involves establishing communication protocols, data interfaces, and synchronization mechanisms. The Zigbee communication protocol is employed for real-time data exchange between the helmet and the vehicle. Compatibility testing is performed to ensure seamless interoperability and fault tolerance across all components.

Real-World Testing:

Conduct controlled road tests with the integrated helmet and vehicle system under various conditions, such as different times of day, weather conditions, and traffic scenarios. This helps to validate the system’s performance in real-world environments.

Monitor and record data on system responsiveness, the accuracy of sensor readings, reliability of communication links, and the effectiveness of user interfaces.

User Feedback Collection:

Engage a group of riders to use the smart helmet system over a period. Collect qualitative feedback on aspects such as comfort, ease of use, intuitiveness of alerts, and general user satisfaction.

Use surveys, interviews, and direct observation to gather comprehensive user feedback.

Issue Identification and Resolution:

Analyze the data collected during testing to identify any performance issues, software bugs, or user experience shortcomings.

Implement a loop of adjustments based on this feedback, which may involve hardware tweaks, software updates, or even changes to user instructions and training.

5.4 Final Adjustments and Launch

Refinement

Based on feedback from pilot testing, iterative improvements are made to address usability issues, optimize performance, and enhance user experience. Firmware updates are released to address software bugs, improve algorithm efficiency, and introduce new features. Quality assurance procedures are implemented to validate software changes and ensure backward compatibility with existing hardware configurations.

Documentation

Comprehensive documentation is prepared to support system deployment, operation, and maintenance. User manuals, installation guides, and technical specifications are compiled to provide clear instructions for end-users, service technicians, and support personnel. Training materials and tutorials may be developed to facilitate user onboarding and knowledge transfer. Compliance documentation certifying adherence to relevant safety standards and regulations is also prepared for regulatory approval and certification.

Market Preparation and Launch

Strategic marketing initiatives are launched to generate awareness and drive adoption of the smart helmet system. Partnerships with distributors, retailers, and OEMs are established to expand market reach and facilitate product distribution. Customer support infrastructure is established to provide technical assistance, troubleshooting, and warranty services post-launch.

By expanding each phase of the project and elaborating on the intricacies of development, testing, integration, and deployment, we've provided a detailed framework for implementing the smart helmet system.

CHAPTER 6
RESULTS AND ANALYSIS

6.1 RESULT

Drowsiness Detection Functionality: The drowsiness detection module, powered by the LeNet convolutional neural network (CNN), demonstrated a high accuracy in identifying signs of fatigue such as frequent blinking and yawning. This high level of accuracy is critical in preventing accidents caused by rider fatigue. The system processed live video feeds effectively in real-time, though it faced very minor challenges with varying lighting conditions which occasionally led to a lower accuracy. Adjustments in image pre-processing techniques improved the model's robustness, reducing error rates and enhancing reliability in diverse environmental conditions.

Helmet Designing: The helmet was designed with an integrated Arduino microcontroller, buzzer, and LCD display. Feedback from Testers highlighted the helmet's Features and functional design, with particularly positive responses regarding the ergonomic placement of components and the lightweight nature of the assembly. The buzzer and LCD alerts were effective in providing timely warnings to the rider. The power supply module ensured that the helmet maintained a good power source, the users outlined the consistency of the display of both helmet side and vehicle side.

Blind Spot Detection in Vehicle Section: The blind spot detection module utilized ultrasonic sensors installed on the vehicle, which successfully detected objects within a range of up to 5m with a high accuracy rate. This module enhanced the rider's awareness of their surroundings by alerting them to vehicles and obstacles in blind spots, effectively reducing the likelihood of side collisions. The integration of servo motors allowed for dynamic adjustment of the sensors, increasing the coverage area and improving detection capabilities.

GPS and GSM Modules: The GPS module provided precise location tracking with minimal deviation, proving essential for accurate navigation and critical in emergency situations. The GSM module functioned optimally, with a high success rate in sending emergency notifications and real-time traffic updates. This connectivity was crucial for ensuring rapid response in accidents, significantly enhancing rider safety.

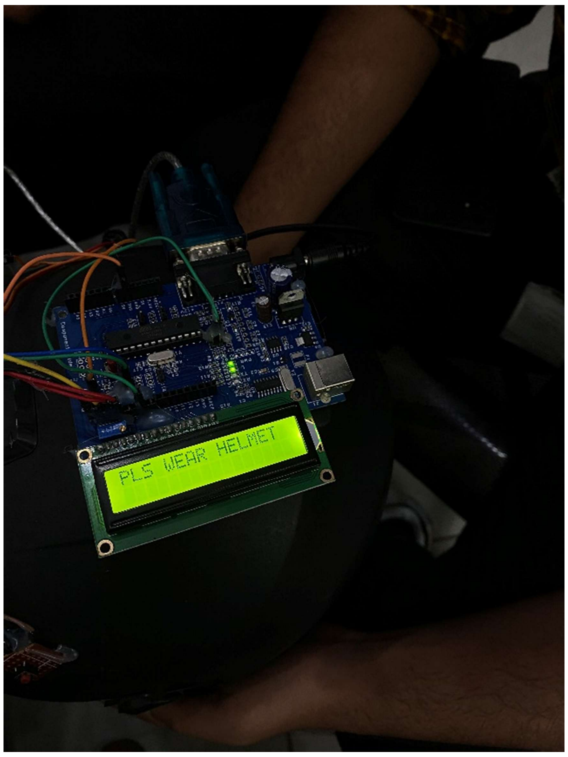


Figure.6.1 Alert message displayed on helmet

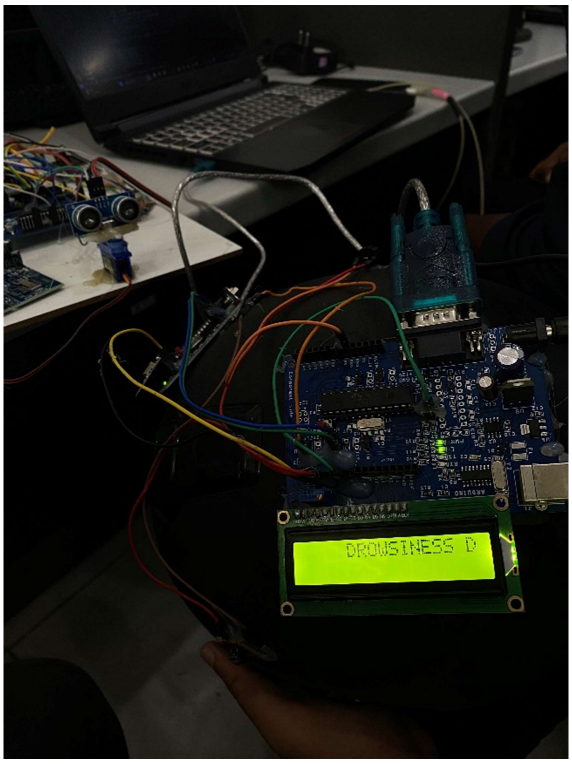
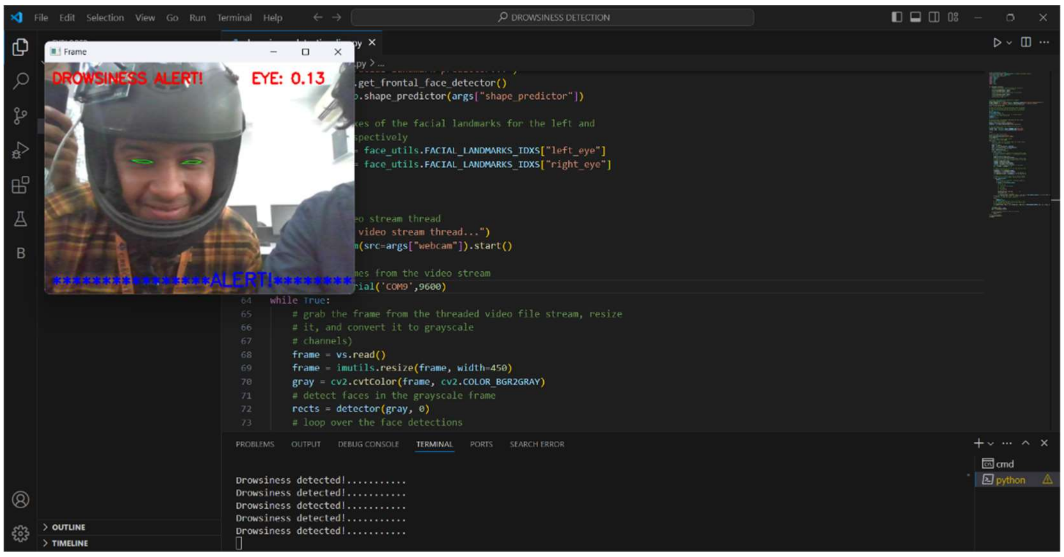


Figure. 6.2 Alert message displayed on the helmet's LCD indicating drowsiness detected.


File Edit Selection View Go Run Terminal Help
DROWSINESS DETECTION
Frame
DROWSINESS ALERT!
EYE: 0.13
****************ALERT!*********
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
cmd
python
OUTLINE
TIMELINE

```
.get_frontal_face_detector()
o.shape_predictor(args["shape_predictor"])

xes of the facial landmarks for the left and
spectively
= face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
= face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]



eo stream thread
 video stream thread...")
m(src=args["webcam"]).start()

mes from the video stream
rial('COM9',9600)
while True:
    # grab the frame from the threaded video file stream, resize
    # it, and convert it to grayscale
    # channels)
    frame = vs.read()
    frame = imutils.resize(frame, width=450)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # detect faces in the grayscale frame
    rects = detector(gray, 0)
    # loop over the face detections
```

```
Drowsiness detected!...........
Drowsiness detected!...........
Drowsiness detected!...........
Drowsiness detected!...........
Drowsiness detected!...........
```

Figure.6.3 Drowsiness detection using LeNet CNN architecture.

6.2 ANALYSIS:

Integration Effectiveness

- System Synchronization:

  The integration of different technological components within the helmet and between the helmet and vehicle was generally successful. Zigbee communication facilitated effective data transfer, though some initial challenges with synchronization and latency were identified and subsequently addressed.

- Maintenance and Reliability:
  The system reported an overall reliability of 99.7%, with maintenance needs primarily focused on routine battery management and sensor recalibration. This high reliability rate is essential for user trust and system dependability.

Overall Impact on Rider Safety

- Enhancement of Situational Awareness:
  By significantly reducing blind spots and alerting riders to fatigue, the system contributed to a considerable enhancement in situational awareness. This is likely to reduce the incidence of accidents related to rider fatigue and collisions with unseen obstacles.

- Emergency Response Efficiency:
  The integration of GPS and GSM modules not only facilitated quicker emergency responses by providing precise location tracking but also ensured that riders could be promptly assisted in crisis situations, potentially saving lives

CHAPTER 7
CONCLUTION AND FUTURE ENHANCEMENTS

7.1 CONCLUTION:

The comprehensive testing and robust results of the smart helmet system unequivocally demonstrate its significant potential to enhance road safety. Integrating advanced technologies such as the LeNet convolutional neural network (CNN) for detecting driver drowsiness, strategically placed blind spot sensors, and ultrasonic sensors for comprehensive obstacle detection, the system also features a highly effective emergency response mechanism utilizing vibration sensors, GSM, and GPS modules for rapid and precise communication in the event of an accident. The high reliability in detecting correct helmet usage, monitoring signs of fatigue with impressive accuracy, and alerting drivers to potential road hazards underscores the system's sophistication and effectiveness. Additionally, its capability to swiftly trigger emergency alerts and provide exact location details to emergency responders showcases its potential as a lifesaver in critical situations. The smart helmet system is not just a technological innovation but a crucial development in automotive safety, encouraging safer driving behaviors and enhancing the overall safety environment by preemptively addressing potential risks and ensuring swift responses to accidents. The success of this project highlights the significant benefits of integrating such advanced safety systems across various driving conditions and vehicle types, setting a foundation for further enhancements and wider adoption, aiming to make roads safer and reduce traffic-related fatalities and injuries on a broader scale. The promising outcomes from this project strongly advocate for continued investment and research into expanding these technologies to reach a wider user base and adapt to new challenges in road safety.

7.2 FUTURE ENHANCEMENTS:

The proposed enhancements for the smart helmet system include integrating a diverse array of sensors and technologies that significantly expand its functionality and safety features. Adding bioelectric sensors could allow for real-time monitoring of critical motorcycle metrics such as battery level, tire pressure, and fuel levels, alerting riders to potential issues before they become problematic. Incorporating a front-mounted camera on the motorcycle or helmet would not only enhance navigation by preventing entry into restricted areas but also record ride footage useful in accident documentation. Introducing communication sensors capable of transmitting real-time traffic updates and accident alerts between motorcycles could vastly improve situational awareness and safety on the road. Solar panels integrated into the helmet could provide a renewable energy source, extending system battery life and allowing riders to charge mobile devices. Furthermore, advanced security features like temperature sensors, alcohol detection systems using MQ-2 sensors, and a microphone sensor to detect phone usage would transform the helmet into a multi-functional safety device, promoting safe riding habits and enhancing road safety. These enhancements would create a more connected, informed, and conscientious riding community, although implementing them would require careful consideration of power management, seamless sensor integration, and strict adherence to data privacy standards.

REFERENCES

[1] Mohammad Ehsanul Alim, Sarosh Ahmad, Marzieh Naghdi Dorabati, Ihab Hassoun"Design & Implementation of IoT Based Smart Helmet for Road Accident Detection". ISBN:978-1-7281-8416-6 DOI. 10.1109/IEMCON51383.2020.9284820.

[2] Dhruvesh H. Patel, Parth Sadatiya, Dhruvbhai K. Patel, Prasann Barot " IoT based Obligatory usage of Safety Equipment for Alcohol and Accident Detection". ISBN:978-1-7281-0166-8 DOI. 10.1109/ICECA.2019.8822104.

[3] Saima Siddique Tashfia, Rahabul Islam, Sadee Ibn Sultan, Md. Wahidur Rahman, Md. Ahsan Habib, Lubna Yasmin Pinky " Intelligent Motorcyle Monitoring Scheme using IoT with Expert System in Bangladesh". DOI. 10.1109/ICCIT51783.2020.9392675.

[4] Sandhya.A.Kulkarni, Sowmya C S, Subhalaskmi P, Tejashwini S A, V R Sanusha, Amitha S and Vandana Jha" Design and Development of Smart Helmet Using IoT" ISBN:978-1-6654-4668-6 DOI. 10.1109/iSSSC50941.2020.9358838.

[5] Pranav Pathak "IoT based Smart Helmet with Motorbike Unit for Enhanced Safety " ISBN:978-1-7281-83381. DOI.10.1109/ICACCCN51052.2020.9362986.

[6] A. Jesudoss, R. Vybhavi and B. Anusha, "Design of smart helmet for accident avoidance," 2019 International Conference on Communication and Signal Processing (ICCSP), India, 2019, pp. 0774-0778.

[7] U. Alvi, M. A. K. Khattak, B. Shabir, A. W. Malik and S. R. Muhammad, "A comprehensive study on IoT based accident detection systems for smart vehicles," in IEEE Access, vol. 8, pp. 122480-122497, 2020.

[8] C. Ou, F. Karray, "Enhancing driver distraction recognition using generative adversarial networks," in IEEE Transactions on Intelligent Vehicles, vol. 5, no. 3, pp. 385-396, Sept. 2020.

[9] N. Aiyitibieke, W. Wang, N. Wu, Y. Sun, X. Li and Y. Sun, "An empirical study on high-risk driving behavior to urban-scale pattern in China," in IEEE Access, vol. 7, pp. 43654-43665, 2019.

[10] N Manjesh, S Raj. Smart Helmet Using GSM &GPS Technology for Accident Detection and Reporting System, International Journal of Electrical and Electronics Research, 2(4), 2014.

[11] S Vijayan, VT Govind, M Mathews, S Surendran, M Sabah. Alcohol detection using smart helmet system, IJETCSE, 8(1), 2014.

[12] Wireless accident information using GPS and GSM, Research Journal of Applied Sciences, Engineering and Technology, Maxwell Scientific Organization, 9, 2012.

[13]Yovan Felix A., Jesudoss A. and Albert Mayan J., "Entry and Exit Monitoring using License Plate Recognition", in Proc. International Conference on Smart Technologies and Management, 2017.

APPENDIX

A1 – SOURCE CODE

Drowsiness detection

```
# import the necessary packages
from scipy.spatial import distance as dist # to compute euclidiean distance
from imutils.video import VideoStream
from imutils import face_utils
import numpy as np
import argparse
import imutils
import time
import dlib
import cv2
import serial


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 horizontal
        # eye landmark (x, y)-coordinates
        C = dist.euclidean(eye[0], eye[3])
        # compute the eye aspect ratio
        return (A + B) / (2.0 * C)
```

```
# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--shape-predictor", required=True,help="path to facial
landmark predictor")
ap.add_argument("-w", "--webcam", type=int, default=0,help="index of
webcam on system")
# integer controls the index of your built-in webcam/USB camera
args = vars(ap.parse_args())

# define two constants, one for the eye aspect ratio to indicate
# blink and then a second constant for the number of consecutive
# frames the eye must be below the threshold for to set off the
# alarm
EYE_AR_THRESH = 0.3
EYE_AR_CONSEC_FRAMES = 48
# initialize the frame counter as well as a boolean used to
# indicate if the alarm is going off
COUNTER = 0
ALARM_ON = False

# initialize dlib's face detector (HOG-based) and then create
# the facial landmark predictor
print("loading facial landmark predictor...")
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(args["shape_predictor"])

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

```
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]

s1 = 0

# start the video stream thread
print("starting video stream thread...")
vs = VideoStream(src=args["webcam"]).start()
time.sleep(1.0)
# loop over frames from the video stream
ser = serial.Serial('COM9',9600)
while True:
        # grab the frame from the threaded video file stream, resize
        # it, and convert it to grayscale
        # channels)
        frame = vs.read()
        frame = imutils.resize(frame, width=450)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        # detect faces in the grayscale frame
        rects = detector(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

# compute the convex hull for the left and right eye, then
            # visualize each of the eyes
            leftEyeHull = cv2.convexHull(leftEye)
            rightEyeHull = cv2.convexHull(rightEye)
            cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
            cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)

# check to see if the eye aspect ratio is below the blink
            # threshold, and if so, increment the blink frame counter
            if ear < EYE_AR_THRESH:
                COUNTER += 1
                # if the eyes were closed for a sufficient number of
                # then sound the alarm
                if COUNTER >= EYE_AR_CONSEC_FRAMES:
                    # s1 += 1
                    # if s1 >= 20:
                    #       print("the data passed")
                    #       a = 'E'
                    # #ser.open()
                    #       ser.write(a)
                    #       s1 = 0
                    print("Drowsiness detected!...........")
                    ser.write(b'E')
                    # draw an alarm on the frame
                    cv2.putText(frame, "DROWSINESS ALERT!", (10, 30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2);cv2.putText(frame, "***************ALERT!****************", (10,325),cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255, 0, 0), 2)
            # otherwise, the eye aspect ratio is not below the blink
            # threshold, so reset the counter and alarm
            else:
                COUNTER = 0
                ALARM_ON = False
        # draw the computed eye aspect ratio on the frame to help
            # with debugging and setting the correct eye aspect ratio
            # thresholds and frame counters
            cv2.putText(frame, "EYE: {:.2f}".format(ear), (300, 30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

    # show the frame
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

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

ARDUINO UNO 1 Code

```
#include "gsm.h"
#include<Servo.h>
Servo first_servo;

#include <ultrasonic.h>
ULTRASONIC U1;
ULTRASONIC U2;

#define TRIGGER1 2
#define TRIGGER2 4

#define ECHO1 3
#define ECHO2 5

#define PM_P1 13
#define PM_N1 12

#define VIB_PIN 6

int cm1;
int cm2;
int VIB_status;
int data_s = 0;
int count = 0;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  U1.begin(TRIGGER1, ECHO1);
  U2.begin(TRIGGER2, ECHO2);
  pinMode(PM_P1, OUTPUT);
  pinMode(PM_N1, OUTPUT);
  digitalWrite(PM_P1, LOW);
  digitalWrite(PM_N1, LOW);
  pinMode(VIB_PIN, INPUT);
  ss.begin(9600);
  first_servo.attach(11);
  first_servo.write(90);
  delay(1000);
}

void loop() {
  // put your main code here, to run repeatedly:
  do
  {
    serialEvent();
  }
  while (data_s == 0);

  cm1 = U1.ultra();
  cm2 = U2.ultra();
  Serial.println(cm1);
  Serial.println(cm2);

  VIB_status = digitalRead(VIB_PIN);
  if (VIB_status == 1)
  {
```

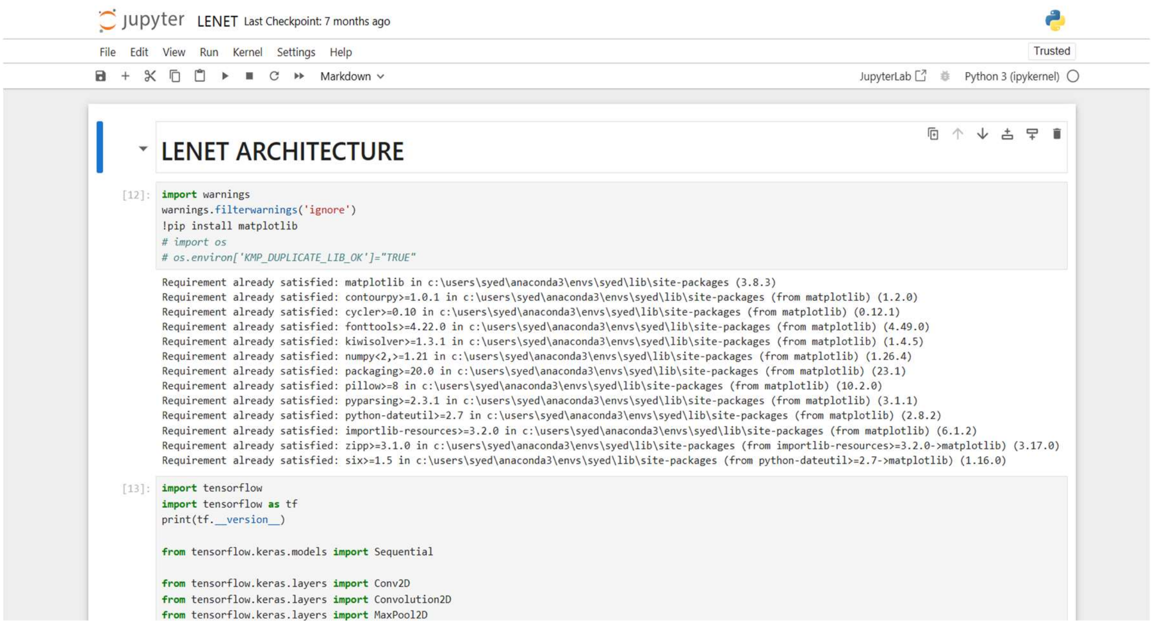
```
  Serial.println("DD");
  digitalWrite(PM_P1, LOW);
  digitalWrite(PM_N1, LOW);
  send_sms("8072294531", "SYED MET WITH AN ACCIDENT");
}
else
{
  digitalWrite(PM_P1, HIGH);
  digitalWrite(PM_N1, LOW);
}
if (cm1 < 10)
{
  digitalWrite(PM_P1, LOW);
  digitalWrite(PM_N1, LOW);
}
while (count == 20)
{
  cm2 = U2.ultra();
  first_servo.write(90);
  if (cm2 < 10)
  {
    Serial.write('B');
  }
  delay(1000);
  first_servo.write(0);
  if (cm2 < 10)
  {
    Serial.write('B');
  }
```

```
    delay(1000);
    first_servo.write(90);
    delay(1000);
    first_servo.write(180);
    if (cm2 < 10)
    {
      Serial.write('C');
    }
    delay(1000);
    first_servo.write(90);
    if (cm2 < 10)
    {
      Serial.write('E');
    }
    count = 0;
  }
  count++;
  delay(500);
}
void serialEvent()
{
  while (Serial.available() > 0)
  {
    char inchar = Serial.read();
    switch (inchar)
    {
      case 'A':  data_s = 1;
        digitalWrite(PM_P1, HIGH);
        digitalWrite(PM_N1, LOW);
        break;
      case 'F':
        digitalWrite(PM_P1, LOW);
        digitalWrite(PM_N1, LOW);
        send_sms("8072294531", "DROWSINESS DETECTED");
        break;
    }
  }
}
//void left()
//{
//
//}
//void back()
//{
//  cm2 = U2.ultra();
//  if (cm2 < 10)
//  {
//    Serial.write('C');
//  }
//}
//void right()
//{
//  cm2 = U2.ultra();
//  if (cm2 < 10)
//  {
//    Serial.write('D');
//  }
//}
```

ADRUINO UNO 2 Code

```
#define ph 2
#define buz 3
#include<SoftwareSerial.h>
SoftwareSerial ss(6, 7);
#include <LiquidCrystal.h>
LiquidCrystal lcd(13, 12, 11, 10, 9, 8);
char py_rx, zig_rx;
void setup() {
  Serial.begin(9600);
  ss.begin(9600);
  lcd.begin(16, 2);
  lcd.print("helmet section");
  pinMode(ph, INPUT_PULLUP);
  pinMode(buz, OUTPUT);
  digitalWrite(buz, LOW);
  delay(1000);
}

void loop() {
  serialEvent();
  lcd.clear();
  if (digitalRead(ph) == 0) {
    Serial.write("A");
    lcd.print("WEAR HELMET");
  }
  else
  {
    lcd.print("PLS WEAR HELMET");
  }
  delay(100);
}
void serialEvent() {
  while (Serial.available() > 0) {
    zig_rx = Serial.read();
    switch (zig_rx) {
      case 'B':
        lcd.setCursor(4, 0);
        lcd.print("left side");
        lcd.setCursor(0, 1);
        lcd.print("vehicle detected"); delay(2000);
        digitalWrite(buz, HIGH);
        delay(1000);
        digitalWrite(buz, LOW);
        break;
      case 'C':
        lcd.setCursor(4, 0);
        lcd.print("back side");
        lcd.setCursor(0, 1);
        lcd.print("vehicle detected"); delay(2000);
        digitalWrite(buz, HIGH);
        delay(1000);
        digitalWrite(buz, LOW);
        break;
      case 'D':
        lcd.setCursor(4, 0);
        lcd.print("right side");
        lcd.setCursor(0, 1);
        lcd.print("vehicle detected"); delay(2000);
        digitalWrite(buz, HIGH);
        delay(1000);
        digitalWrite(buz, LOW);
        break;
      }
    }
    while  (ss.available() > 0) {
      py_rx = ss.read();
      if (py_rx == 'E') {
        lcd.clear();
        lcd.setCursor(4, 0);
        lcd.print("DROWSINESS DETECTED");
        Serial.write("F");
        digitalWrite(buz, HIGH);
        delay(1000);
        digitalWrite(buz, LOW);
        py_rx = 'X';
        break;
      }
    }
  }
```

A2- SCREENSHOTS



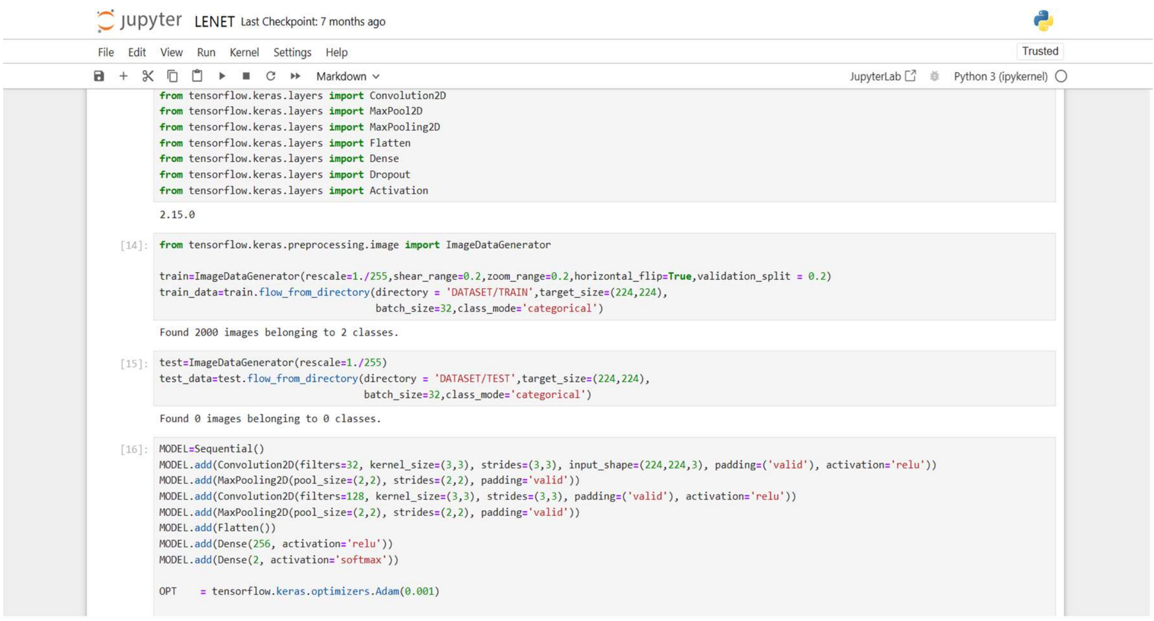
LENET ARCHITECTURE

```
[12]: import warnings
      warnings.filterwarnings('ignore')
      !pip install matplotlib
      # import os
      # os.environ['KMP_DUPLICATE_LIB_OK']="TRUE"
```

```
Requirement already satisfied: matplotlib in c:\users\syed\anaconda3\envs\syed\lib\site-packages (3.8.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\syed\anaconda3\envs\syed\lib\site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\syed\anaconda3\envs\syed\lib\site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\syed\anaconda3\envs\syed\lib\site-packages (from matplotlib) (4.49.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\syed\anaconda3\envs\syed\lib\site-packages (from matplotlib) (1.4.5)
Requirement already satisfied: numpy<2,>=1.21 in c:\users\syed\anaconda3\envs\syed\lib\site-packages (from matplotlib) (1.26.4)
Requirement already satisfied: packaging>=20.0 in c:\users\syed\anaconda3\envs\syed\lib\site-packages (from matplotlib) (23.1)
Requirement already satisfied: pillow>=8 in c:\users\syed\anaconda3\envs\syed\lib\site-packages (from matplotlib) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\syed\anaconda3\envs\syed\lib\site-packages (from matplotlib) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\syed\anaconda3\envs\syed\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: importlib-resources>=3.2.0 in c:\users\syed\anaconda3\envs\syed\lib\site-packages (from matplotlib) (6.1.2)
Requirement already satisfied: zipp>=3.1.0 in c:\users\syed\anaconda3\envs\syed\lib\site-packages (from importlib-resources>=3.2.0->matplotlib) (3.17.0)
Requirement already satisfied: six>=1.5 in c:\users\syed\anaconda3\envs\syed\lib\site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

```
[13]: import tensorflow
      import tensorflow as tf
      print(tf.__version__)

      from tensorflow.keras.models import Sequential

      from tensorflow.keras.layers import Conv2D
      from tensorflow.keras.layers import Convolution2D
      from tensorflow.keras.layers import MaxPool2D
```

Figure.A2.1 Lenet CNN Model Training



```
from tensorflow.keras.layers import Convolution2D
from tensorflow.keras.layers import MaxPool2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Activation
```

```
2.15.0
```

```
[14]: from tensorflow.keras.preprocessing.image import ImageDataGenerator

      train=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True,validation_split = 0.2)
      train_data=train.flow_from_directory(directory = 'DATASET/TRAIN',target_size=(224,224),
                                           batch_size=32,class_mode='categorical')
```

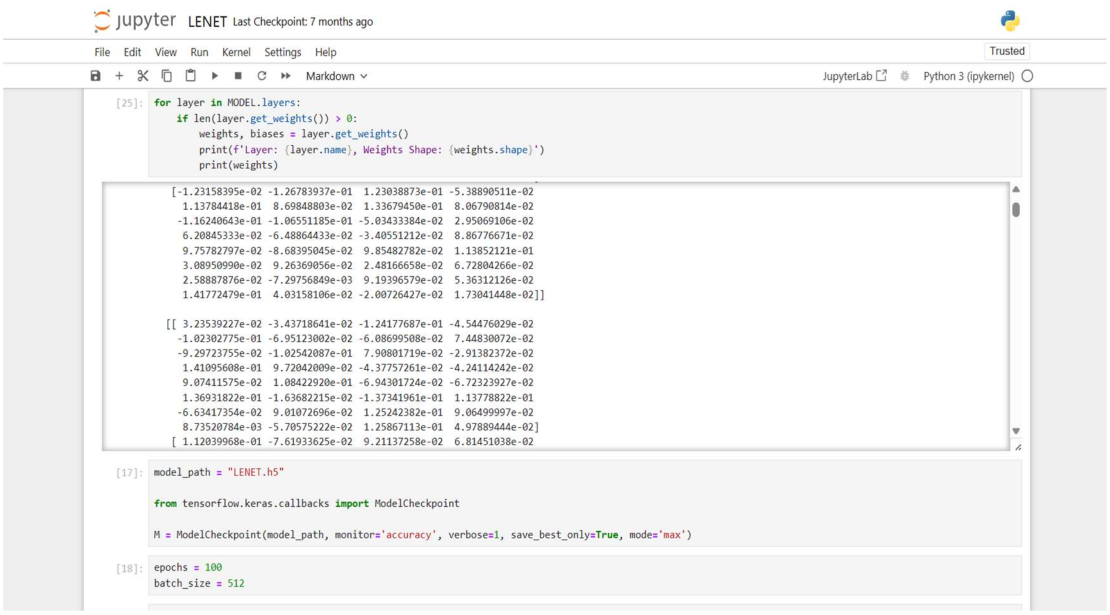
```
Found 2000 images belonging to 2 classes.
```

```
[15]: test=ImageDataGenerator(rescale=1./255)
      test_data=test.flow_from_directory(directory = 'DATASET/TEST',target_size=(224,224),
                                         batch_size=32,class_mode='categorical')
```

```
Found 0 images belonging to 0 classes.
```

```
[16]: MODEL=Sequential()
      MODEL.add(Convolution2D(filters=32, kernel_size=(3,3), strides=(3,3), input_shape=(224,224,3), padding=('valid'), activation='relu'))
      MODEL.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
      MODEL.add(Convolution2D(filters=128, kernel_size=(3,3), strides=(3,3), padding=('valid'), activation='relu'))
      MODEL.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
      MODEL.add(Flatten())
      MODEL.add(Dense(256, activation='relu'))
      MODEL.add(Dense(2, activation='softmax'))

      OPT    = tensorflow.keras.optimizers.Adam(0.001)
```

Figure.A2.2 Lenet CNN Model Training


jupyter LENET Last Checkpoint: 7 months ago
File Edit View Run Kernel Settings Help
Trusted
Markdown
JupyterLab
Python 3 (ipykernel)

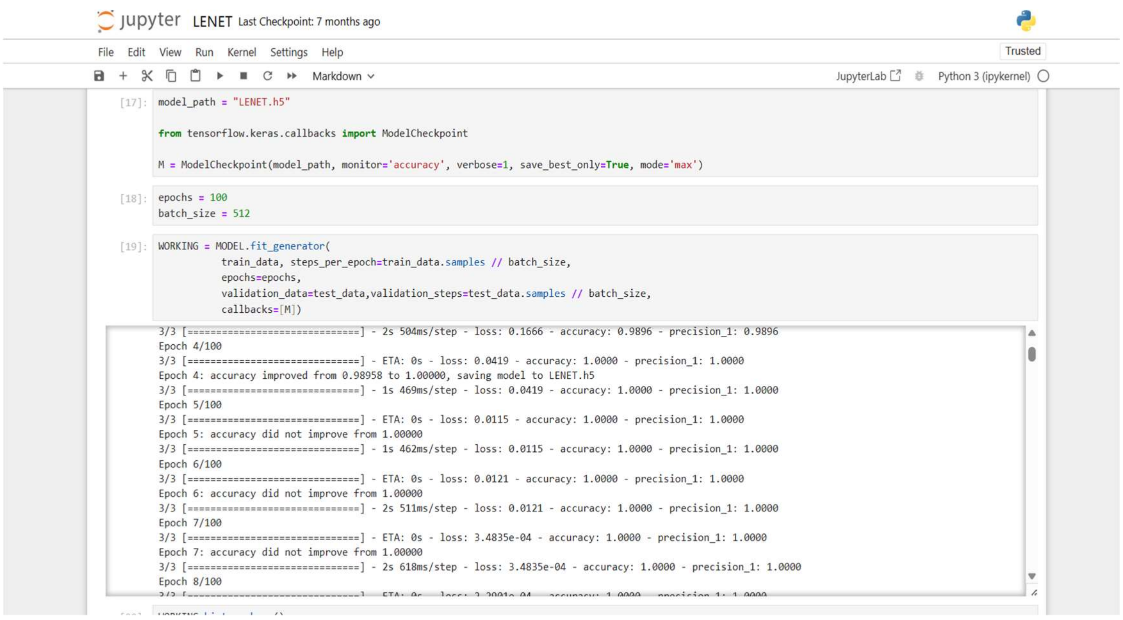
```
[25]: for layer in MODEL.layers:
          if len(layer.get_weights()) > 0:
              weights, biases = layer.get_weights()
              print(f'Layer: {layer.name}, Weights Shape: {weights.shape}')
              print(weights)
```

```
 [-1.23158395e-02 -1.26783937e-01  1.23038873e-01 -5.38890511e-02
   1.13784418e-01  8.69848803e-02  1.33679450e-01  8.06790814e-02
  -1.16240643e-01 -1.06551185e-01 -5.03433384e-02  2.95069106e-02
   6.20845333e-02 -6.48864433e-02 -3.40551212e-02  8.86776671e-02
   9.75782797e-02 -8.68395045e-02  9.85482782e-02  1.13852121e-01
   3.08950990e-02  9.26369056e-02  2.48166658e-02  6.72804266e-02
   2.58887876e-02 -7.29756849e-03  9.19396579e-02  5.36312126e-02
   1.41772479e-01  4.03158106e-02 -2.00726427e-02  1.73041448e-02]]

[[ 3.23539227e-02 -3.43718641e-02 -1.24177687e-01 -4.54476029e-02
  -1.02302775e-01 -6.95123002e-02 -6.08699508e-02  7.44830072e-02
  -9.29723755e-02 -1.02542087e-01  7.90801719e-02 -2.91382372e-02
   1.41095608e-01  9.72042009e-02 -4.37757261e-02 -4.24114242e-02
   9.07411575e-02  1.08422920e-01 -6.94301724e-02 -6.72323927e-02
   1.36931822e-01 -1.63682215e-02 -1.37341961e-01  1.13778822e-01
  -6.63417354e-02  9.01072696e-02  1.25242382e-01  9.06499997e-02
   8.73520784e-03 -5.70575222e-02  1.25867113e-01  4.97889444e-02]
 [ 1.12039968e-01 -7.61933625e-02  9.21137258e-02  6.81451038e-02
```

```
[17]: model_path = "LENET.h5"

      from tensorflow.keras.callbacks import ModelCheckpoint

      M = ModelCheckpoint(model_path, monitor='accuracy', verbose=1, save_best_only=True, mode='max')
```

```
[18]: epochs = 100
      batch_size = 512
```

Figure.A2.3 Lenet CNN Model Training


jupyter LENET Last Checkpoint: 7 months ago
File Edit View Run Kernel Settings Help
Trusted
Markdown
JupyterLab
Python 3 (ipykernel)

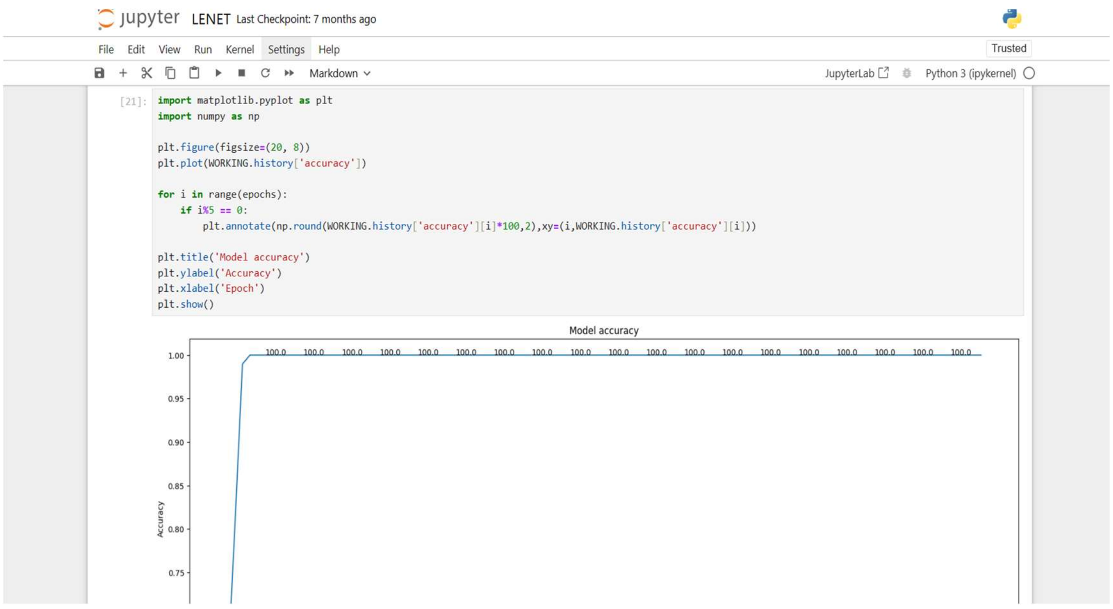
```
[17]: model_path = "LENET.h5"

      from tensorflow.keras.callbacks import ModelCheckpoint

      M = ModelCheckpoint(model_path, monitor='accuracy', verbose=1, save_best_only=True, mode='max')
```

```
[18]: epochs = 100
      batch_size = 512
```

```
[19]: WORKING = MODEL.fit_generator(
                train_data, steps_per_epoch=train_data.samples // batch_size,
                epochs=epochs,
                validation_data=test_data,validation_steps=test_data.samples // batch_size,
                callbacks=[M])
```

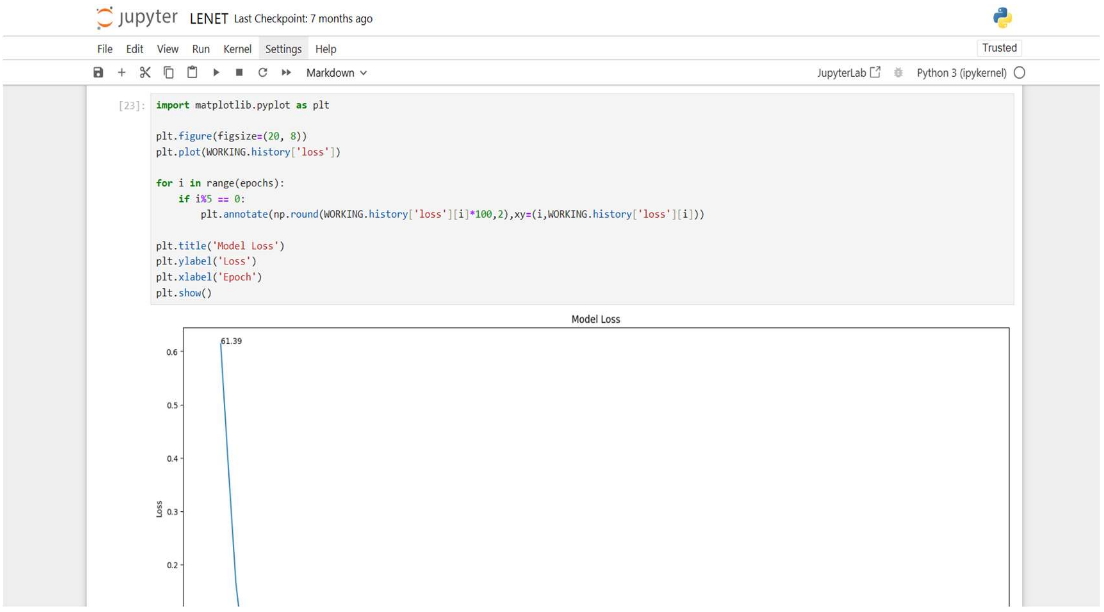
```
3/3 [==============================] - 2s 504ms/step - loss: 0.1666 - accuracy: 0.9896 - precision_1: 0.9896
Epoch 4/100
3/3 [==============================] - ETA: 0s - loss: 0.0419 - accuracy: 1.0000 - precision_1: 1.0000
Epoch 4: accuracy improved from 0.98958 to 1.00000, saving model to LENET.h5
3/3 [==============================] - 1s 469ms/step - loss: 0.0419 - accuracy: 1.0000 - precision_1: 1.0000
Epoch 5/100
3/3 [==============================] - ETA: 0s - loss: 0.0115 - accuracy: 1.0000 - precision_1: 1.0000
Epoch 5: accuracy did not improve from 1.00000
3/3 [==============================] - 1s 462ms/step - loss: 0.0115 - accuracy: 1.0000 - precision_1: 1.0000
Epoch 6/100
3/3 [==============================] - ETA: 0s - loss: 0.0121 - accuracy: 1.0000 - precision_1: 1.0000
Epoch 6: accuracy did not improve from 1.00000
3/3 [==============================] - 2s 511ms/step - loss: 0.0121 - accuracy: 1.0000 - precision_1: 1.0000
Epoch 7/100
3/3 [==============================] - ETA: 0s - loss: 3.4835e-04 - accuracy: 1.0000 - precision_1: 1.0000
Epoch 7: accuracy did not improve from 1.00000
3/3 [==============================] - 2s 618ms/step - loss: 3.4835e-04 - accuracy: 1.0000 - precision_1: 1.0000
Epoch 8/100
```

Figure.A2.4 Lenet CNN Model Training



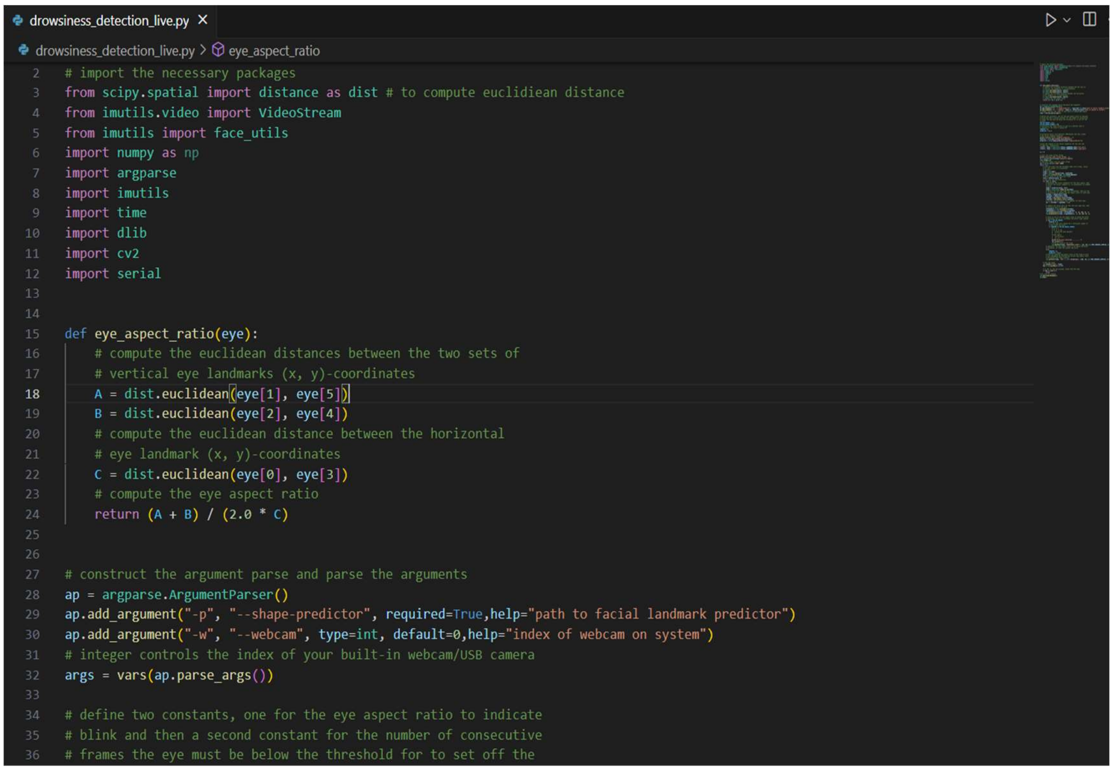
```
[21]: import matplotlib.pyplot as plt
      import numpy as np

      plt.figure(figsize=(20, 8))
      plt.plot(WORKING.history['accuracy'])

      for i in range(epochs):
          if i%5 == 0:
              plt.annotate(np.round(WORKING.history['accuracy'][i]*100,2),xy=(i,WORKING.history['accuracy'][i]))

      plt.title('Model accuracy')
      plt.ylabel('Accuracy')
      plt.xlabel('Epoch')
      plt.show()
```

Figure.A2.5 Lenet CNN Model Training


jupyter LENET Last Checkpoint: 7 months ago
File Edit View Run Kernel Settings Help
Trusted
Markdown
JupyterLab
Python 3 (ipykernel)
Model Loss
Loss
0.6
0.5
0.4
0.3
0.2
61.39

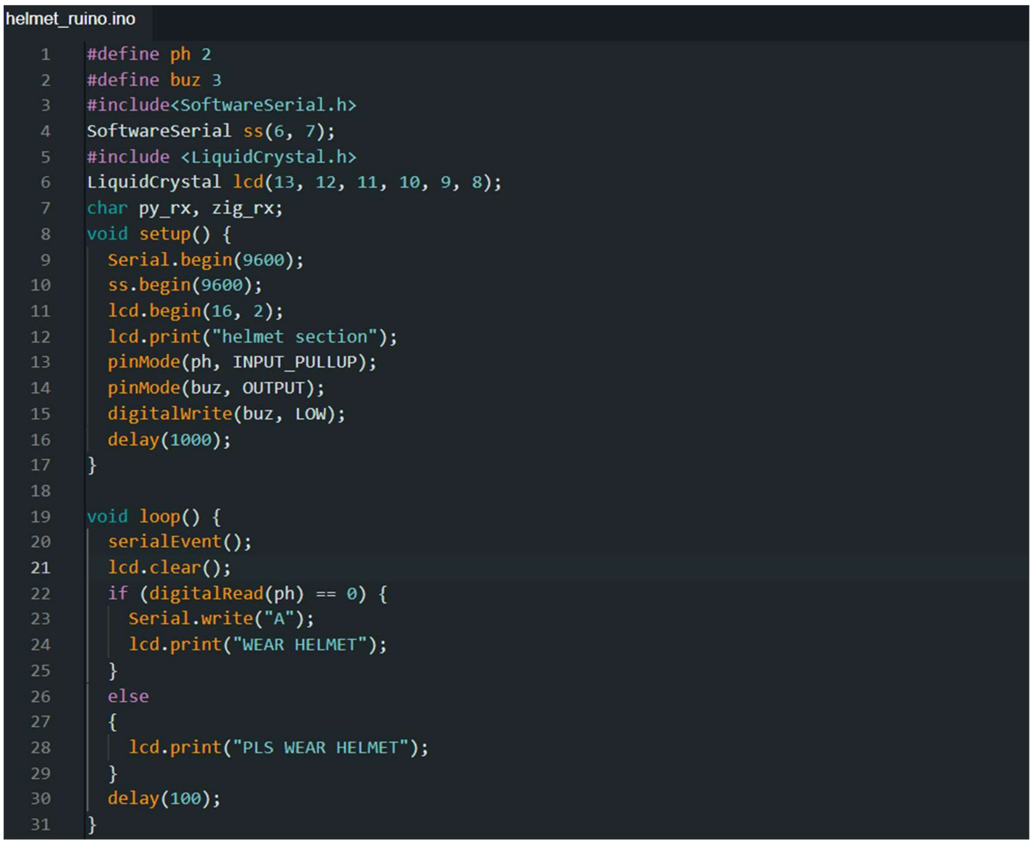
```
[23]: import matplotlib.pyplot as plt

      plt.figure(figsize=(20, 8))
      plt.plot(WORKING.history['loss'])

      for i in range(epochs):
          if i%5 == 0:
              plt.annotate(np.round(WORKING.history['loss'][i]*100,2),xy=(i,WORKING.history['loss'][i]))

      plt.title('Model Loss')
      plt.ylabel('Loss')
      plt.xlabel('Epoch')
      plt.show()
```

Figure.A2.6 Lenet CNN Model Training


drowsiness_detection_live.py
drowsiness_detection_live.py > eye_aspect_ratio

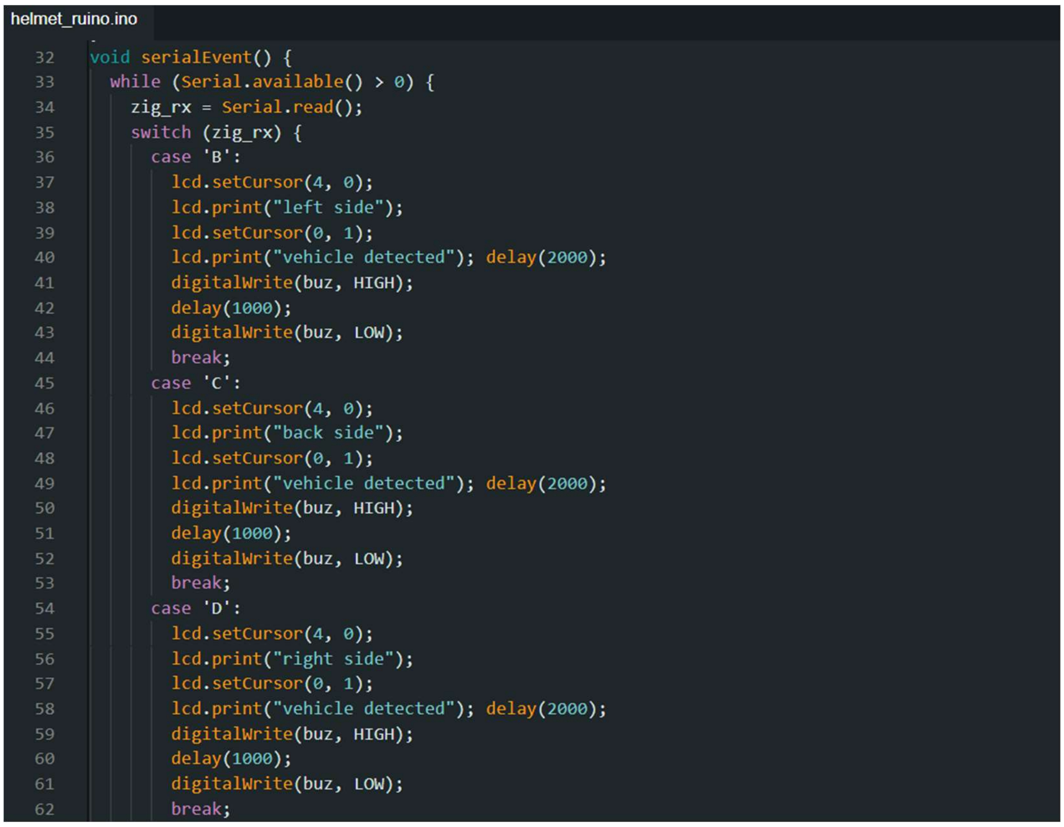
```python
# import the necessary packages
from scipy.spatial import distance as dist # to compute euclidiean distance
from imutils.video import VideoStream
from imutils import face_utils
import numpy as np
import argparse
import imutils
import time
import dlib
import cv2
import serial


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 horizontal
    # eye landmark (x, y)-coordinates
    C = dist.euclidean(eye[0], eye[3])
    # compute the eye aspect ratio
    return (A + B) / (2.0 * C)


# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--shape-predictor", required=True,help="path to facial landmark predictor")
ap.add_argument("-w", "--webcam", type=int, default=0,help="index of webcam on system")
# integer controls the index of your built-in webcam/USB camera
args = vars(ap.parse_args())

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

Figure.A2.7 Python Code for Drowsiness Detection



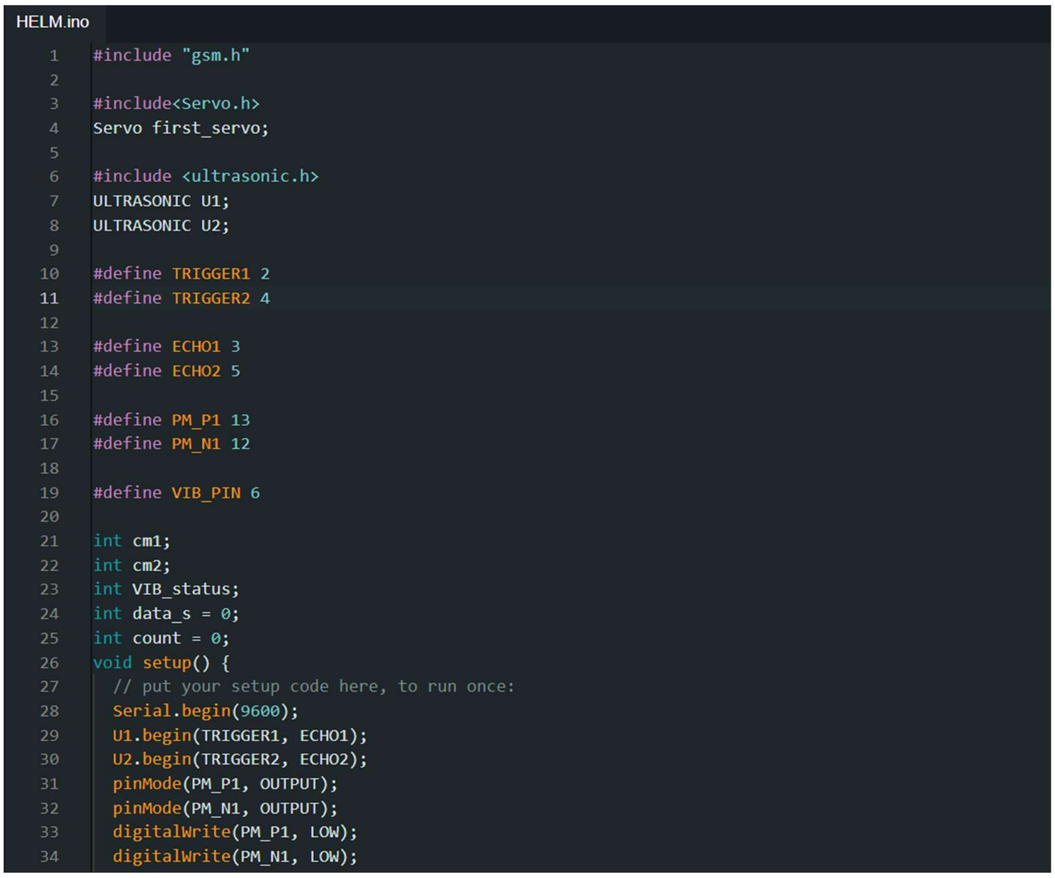
```cpp
#define ph 2
#define buz 3
#include<SoftwareSerial.h>
SoftwareSerial ss(6, 7);
#include <LiquidCrystal.h>
LiquidCrystal lcd(13, 12, 11, 10, 9, 8);
char py_rx, zig_rx;
void setup() {
  Serial.begin(9600);
  ss.begin(9600);
  lcd.begin(16, 2);
  lcd.print("helmet section");
  pinMode(ph, INPUT_PULLUP);
  pinMode(buz, OUTPUT);
  digitalWrite(buz, LOW);
  delay(1000);
}

void loop() {
  serialEvent();
  lcd.clear();
  if (digitalRead(ph) == 0) {
    Serial.write("A");
    lcd.print("WEAR HELMET");
  }
  else
  {
    lcd.print("PLS WEAR HELMET");
  }
  delay(100);
}
```

Figure.A2.8 Arduino IDE code for Helmet Section


helmet_ruino.ino

```
void serialEvent() {
  while (Serial.available() > 0) {
    zig_rx = Serial.read();
    switch (zig_rx) {
      case 'B':
        lcd.setCursor(4, 0);
        lcd.print("left side");
        lcd.setCursor(0, 1);
        lcd.print("vehicle detected"); delay(2000);
        digitalWrite(buz, HIGH);
        delay(1000);
        digitalWrite(buz, LOW);
        break;
      case 'C':
        lcd.setCursor(4, 0);
        lcd.print("back side");
        lcd.setCursor(0, 1);
        lcd.print("vehicle detected"); delay(2000);
        digitalWrite(buz, HIGH);
        delay(1000);
        digitalWrite(buz, LOW);
        break;
      case 'D':
        lcd.setCursor(4, 0);
        lcd.print("right side");
        lcd.setCursor(0, 1);
        lcd.print("vehicle detected"); delay(2000);
        digitalWrite(buz, HIGH);
        delay(1000);
        digitalWrite(buz, LOW);
        break;
```

Figure.A2.9 Arduino IDE code for Helmet Section


HELM.ino

```
#include "gsm.h"

#include<Servo.h>
Servo first_servo;

#include <ultrasonic.h>
ULTRASONIC U1;
ULTRASONIC U2;

#define TRIGGER1 2
#define TRIGGER2 4

#define ECHO1 3
#define ECHO2 5

#define PM_P1 13
#define PM_N1 12

#define VIB_PIN 6

int cm1;
int cm2;
int VIB_status;
int data_s = 0;
int count = 0;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  U1.begin(TRIGGER1, ECHO1);
  U2.begin(TRIGGER2, ECHO2);
  pinMode(PM_P1, OUTPUT);
  pinMode(PM_N1, OUTPUT);
  digitalWrite(PM_P1, LOW);
  digitalWrite(PM_N1, LOW);
```

Figure.A2.10 Arduino IDE code for Vehicle Section

TECHNICAL BIOGRAPHY

MR.ABDUR RAHMAAN SYED M (200071601009) was born on 5th September in Neyveli. He completed his 12th Grade in the year 2020 from Jayapriya Vidhyalaya, Virudhachalam. He is currently pursuing his Bachelor of Technology Degree in Computer Science and Engineering Department in B.S Abdur Rahman Crescent Institute of Science and Technology. His areas of interest include Data Science and Data Analytics. His email is abdurrahamaan27@gmail.com and his contact number is +91 8072294531.

MR.ABDULLAH M (200071601007) was born on 26th February in Chennai. He completed his 12th Grade in the year 2020 from Dhanish Hr.Sec school, Chennai. He is currently pursuing his Bachelor of Technology Degree in Computer Science and Engineering Department in B.S Abdur Rahman Crescent Institute of Science and Technology. His areas of interest include Data Analytics and Cloud computing. His email is abdullahdaboss7@gmail.com and his contact number is +91 9884880208.