

**RAAK COLLEGE OF ENGINEERING AND TECHNOLOGY  
PUDUCHERRY – 605110**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**BONAFIDE CERTIFICATE**

This is to certify that the Project Work titled “**ACCIDENT PREVENTION AT HAIRPIN CURVES USING IoT SENSORS**” is a bonafide record of the work done by **ROGAN. M (21TD0725), SHARON SAJI GEORGE (21TD0727) and THIRUVALLURU SUJITH (21TD0733)** in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** of the **Pondicherry University** during the Academic year 2024- 2025.

**Project Guide**

**Mrs. D. THAMIZHISAI, M.Tech. DCS**

Assistant professor  
(Department of Computer  
Science and Engineering)

**Head of the Department**

**Mr. D. TAMISELVAN, M.Tech. (Ph.D)**

Head of the Department  
(Department of Computer  
Science and Engineering)

Submitted for the University Examination held on\_\_\_\_\_.

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

With immense pleasure, we would like to place on record our thanks to all those who have contributed to the successful completion of this project work.

We are duly bound to express our deep sense of gratitude to our honorable Chairman **Janab Er. B. Mohamed Farouk B.E.**, for his tremendous support and encouragement by providing us a better learning environment with laboratory facilities and the entire college infrastructure to equip ourselves.

We deem it a privilege to record our gratefulness to our beloved Principal/ director **Dr. A. Ravichandran M.E., Ph.D.**, for his unflinching support and consistent encouragement. He has constantly motivated us to remain focused on achieving this project work magnificently.

We are grateful to **Mr. D. Tamilselvan, M.Tech., (Ph.D)**, Associative professor & Head of the Department, Department of **Computer Science and Engineering** who taught us to think ahead and encouraged us to remain focused with the project investigation in depth.

We would like to express our pleasure to the thankful guide **Mrs. D. Thamizhisai, M.Tech., DCS, Project Guide**, Department of Computer Science and Engineering, for her valuable guidance and encouragement throughout the project.

We thank all the faculty members of Department of Computer Science & Engineering, Parents and our friends for helping us to complete the project work successfully in time.

**ROGAN. M** (21TD0725)

**SHARON SAJI GEORGE** (21TD0727)

**THIRUVALLURU SUJITH** (21TD0733)

## TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ACKNOWLEDGEMENT	<i>i</i>
	ABSTRACT	<i>v</i>
	LIST OF FIGURES	<i>vi</i>
	LIST OF ABBREVIATIONS	<i>vii</i>
1	INTRODUCTION	
	1.1 OVERVIEW	01
	1.2 CHALLENGES IN DOMAIN	02
	1.3 OBJECTIVE AND SCOPE OF PROPOSED WORK	03
	1.3.1 OBJECTIVE	
	1.3.2 NEED / SCOPE FOR THE PROJECT WORK	
	1.4 DEEP LEARNING	04
	1.4.1 DEEP LEARNING vs MACHINE LEARNING	
2	LITERATURE SURVEY	
	2.1 SURVEY OF THE RELATED WORKS	07
	2.1.1 IoT BASED SMART SYSTEM FOR DETECTION	
	2.1.2 INTELLIGENT ACCIDENT PREVENTION AND DETECTION	
	2.1.3 ACCIDENT DETECTION AND PREVENTION SYSTEM TO REDUCE TRAFFIC	
	2.2 SUMMARY OF LITERATURE REVIEW	12
3	EXISTING SYSTEM	
	3.1 EXISTING WORK	15
	3.2 PROBLEM FACED IN EXISTING SYSTEM	16

<b>4</b>	<b>PROPOSED SYSTEM</b>	
	4.1 INTRODUCTION	18
	4.2 PROPOSED WORK	19
	4.3 SYSTEM REQUIREMENTS	24
	4.4 PROPOSED ARCHITECTURE DIAGRAM	26
	4.5 ARCHITECTURE DIAGRAM FOR NODE MCU	27
	4.6 WORKING OF PROPOSED SYSTEM	28
	4.6.1 CAMERA FEEDS AND OBJECT DETECTION	
	4.6.2 DATA PROCESSING AND COMPRESSION	
	4.6.3 FIREBASE INTEGRATION FOR DATA STORAGE	
	4.6.4 NODE MODULE AND IoT CONNECTIVITY	
	4.6.5 WORK – FLOW ARCHITECTURE	
	4.7 PERFORMANCE ANALYSIS	30
<b>5</b>	<b>PROPOSED SYSTEM ALGORITHM</b>	
	5.1 YOLO V08 ALGORITHM	32
	5.2 PSEUDO CODE	34
<b>6</b>	<b>ALGORITHM AND TECHNOLOGY</b>	
	6.1 NLP (NATURAL LANGUAGE PROCESSING)	39
	6.2 CNN (CONVOLUTIONAL NEURAL NETWORK)	40
	6.3 AI (ARTIFICIAL INTELLIGENCE)	40
<b>7</b>	<b>MODULE AND MODULE DESCRIPTION</b>	
	7.1 MODULE CLASSIFICATION	41
	7.2 MODULE DESCRIPTION	45
	7.2.1 VIDEO FEED MODULE	
	7.2.2 VEHICLE DETECTION MODULE	
	7.2.3 YOLO ALGORITHM MODULE	
	7.2.4 IOT MODULE	

	7.2.5 NODE MCU MODULE	
<b>8</b>	<b>PROPOSED SYSTEM ADVANTAGE</b>	
	8.1 ADVANTAGE OF THE PROPOSED SYSTEM	50
<b>9</b>	<b>CODING</b>	
	9.1 APP. PY	51
	9.1.1 LIBRARIES & MODEL INITIALIZATION	
	9.1.2 VEHICLE CLASSIFICATION	
	9.1.3 ESP32 – CAM INTEGRATION	
	9.1.4 FRAME PROCESING	
	9.1.5 STREAMING LOGIC	
	9.1.6 FLASK ROUTES	
	9.2 ARDUINO. IDE	59
	9.2.1 WIFI CONNECTION	
	9.2.2 WEB SERVER SETUP	
	9.2.3 TRAFFIC CONTROL LIGHT	
	9.2.4 LCD DISPLAY	
<b>10</b>	<b>SCREENSHOTS</b>	
	10.1 SCREENSHOTS OF OUTPUTS	68
<b>11</b>	<b>CONCLUSION</b>	<b>64</b>
<b>12</b>	<b>REFERENCES</b>	<b>65</b>

## **ABSTRACT**

In today's technologically advanced world, where smartphones play a pivotal role in daily life, leveraging technology for safety and accident prevention becomes crucial. This proposed system addresses the issue of frequent accidents in hilly terrains and ghats by utilizing a combination of computer vision, IoT, and real-time data processing. The system comprises a prototype environment simulating mountainous regions, equipped with two cameras capturing video feeds of the road. These feeds are processed and compressed, serving as input for the Vehicle Detection System, which utilizes YOLO (You Only Look Once) for efficient object detection, focusing on identifying vehicles (toy cars) on the road. Upon detecting a vehicle, the system uploads the relevant data to a Firebase server, which securely stores the information. Simultaneously, a NodeMCU microcontroller retrieves the data from Firebase, triggering a signal on a pole, represented by a red light, indicating the presence of a vehicle on the other side. The entire system operates on an IoT framework, incorporating advanced end-to-end encryption to ensure data integrity and security, preventing any tampering. The chosen network protocol for communication within the system is IoT, facilitating seamless data transfer between components. The real-time database service provided by Google Firebase is integrated into the system, offering efficient data upload and retrieval capabilities. This integration enhances the responsiveness of the system, ensuring timely alerts and updates based on the detected vehicles. In summary, this proposed Intelligent Mountainous Region Traffic Monitoring and Alert System leverages cutting-edge technologies to enhance road safety in challenging terrains.

## LIST OF FIGURES

FIGURE NO.	NAME OF THE FIGURE	PAGE NO.
1.1	Machine Learning vs Deep Laerning	05
1.2	Node MCU	25
1.3	Architecture Diagram	26
1.4	Architecture Diagram of Node MCU	27
1.5	Workflow Architecture	29
1.6	Performance Analysis	31
1.7	Node MCU Chipset	43

## LIST OF ABBREVIATIONS

- **YOLO** - You Only Look Once
- **IoT** - Internet of Things
- **AI** - Artificial Intelligence
- **CV** - Computer Vision
- **RTP** - Real-Time Processing
- **VDS** - Vehicle Detection System
- **RTDB** - Real-Time Database
- **MCU** - Microcontroller Unit
- **HTTP** - Hypertext Transfer Protocol
- **HTTPS** - Hypertext Transfer Protocol Secure
- **API** - Application Programming Interface
- **PWM** - Pulse Width Modulation
- **URL** - Uniform Resource Locator
- **SSL** - Secure Sockets Layer
- **MQTT** - Message Queuing Telemetry Transport
- **LED** - Light Emitting Diode
- **GPS** - Global Positioning System
- **RTOS** - Real-Time Operating System
- **LAN** - Local Area Network
- **RTSP** - Real-Time Streaming Protocol



# **ACCIDENT PREVENTION AT HAIRPIN CURVES USING IOT SENSORS**

## **CHAPTER 1**

### **INTRODUCTION**

#### **1.1OVERVIEW:**

In our rapidly advancing world, the escalating demands of work life have made traffic congestion an increasingly frustrating issue. The primary cause of most traffic jams lies in accidents and vehicle management problems at traffic signals. Addressing this challenge, particularly in hazardous terrains like mountains, is crucial for minimizing accidents and saving lives. The proposed system offers a solution by leveraging advanced technology to enhance road safety.

One significant contributor to accidents in mountainous areas is the lack of awareness regarding approaching vehicles from different turns. To mitigate this risk, the system employs Artificial Intelligence (AI) technology. The AI system is designed to detect accidents promptly and, in the event of an incident, automatically generate an email containing the accident's location. This email is then swiftly sent to the nearest police stations and hospitals.

The use of AI in accident detection significantly reduces the response time, providing a rapid and efficient means of alerting authorities and emergency services. This quick dissemination of information holds the potential to increase the chances of saving victims' lives and minimizing the severity of injuries. By integrating technology to enhance awareness and response mechanisms, the proposed system aims to contribute to a safer and more efficient traffic management system, ultimately reducing accidents and alleviating the impact of traffic jams on people's lives.

## **1.2 CHALLENGES IN DOMAIN:**

In the realm of IoT and AI, while there are significant advancements in enhancing safety and efficiency, there are also inherent challenges that must be addressed. One primary challenge lies in the interoperability and standardization of devices within the Internet of Things (IoT). The diverse array of sensors, protocols, and communication technologies often leads to compatibility issues, making it challenging to seamlessly integrate devices from different manufacturers or operating on different platforms. This interoperability challenge can hinder the creation of a cohesive and interconnected IoT ecosystem. Furthermore, the integration of AI in IoT systems introduces complexities related to data privacy and security.

As AI relies heavily on vast datasets for training and decision-making, ensuring the confidentiality and integrity of this data becomes crucial. The potential for cyber-attacks and unauthorized access to sensitive information poses a significant challenge, necessitating robust security measures and encryption protocols to safeguard the interconnected devices and the data they generate.

Moreover, the deployment of AI algorithms in real-time applications, such as accident detection in smart traffic systems, demands computational resources and low-latency communication. Balancing the need for sophisticated AI capabilities with the limitations of edge devices in terms of processing power and energy consumption presents a significant challenge in designing efficient and responsive IoT-AI solutions. Addressing these challenges requires a holistic approach, involving standardized protocols for device communication, robust cybersecurity measures, and optimization of AI algorithms for edge computing. Overcoming these obstacles will pave the way for more seamless integration of IoT and AI technologies, unlocking their full potential in creating intelligent and responsive systems for various domains, including traffic management and safety.

## **1.3 OBJECTIVE AND SCOPE OF PROPOSED WORK**

### **1.3.1 OBJECTIVE:**

The primary objective of our project is to enhance safety measures and driving independence in hill areas, specifically focusing on accident prevention and detection in turns. Leveraging the Internet of Things (IoT) and Machine Learning (ML), our system aims to proactively address the challenges posed by accidents in these terrains. In the event of an accident, our solution employs the Global Positioning System (GPS) and Global System for Mobile Communication (GSM) technologies to swiftly alert nearby hospitals, police stations, or rescue teams, facilitating a rapid response and potentially saving lives.

The project also prioritizes driver awareness in challenging terrains like ghats. To achieve this, the system utilizes advanced technologies, including ML algorithms, to detect and notify drivers about approaching vehicles. By implementing this aspect of the project, we aim to empower drivers with real-time information, reducing the risk of accidents caused by unawareness in turns. Ultimately, our objective is to create a comprehensive solution that not only prevents and detects accidents but also promotes a safer driving environment, especially in hilly regions, contributing to overall road safety and emergency response effectiveness.

### **1.3.2 NEED/SCOPE FOR THE PROJECT WORK:**

The need and scope for the proposed project are significant in addressing the pressing issues associated with road safety in hilly terrains, particularly in turns. In these areas, the risk of accidents is heightened due to the challenging topography and limited visibility around bends. The project's primary focus on leveraging Internet of Things (IoT) and Machine Learning (ML) technologies to prevent and detect accidents aligns with the growing demand for innovative solutions in the field of transportation safety.

The scope of the project extends beyond accident prevention to encompass a comprehensive system that integrates GPS and GSM technologies. By incorporating these tools, the project seeks to expedite emergency responses by notifying nearby hospitals, police stations, or rescue teams in the event of an accident, thereby increasing the chances of prompt intervention and reducing potential fatalities or injuries.

Furthermore, the project's emphasis on enhancing driver awareness by using ML algorithms to detect approaching vehicles in ghats broadens its scope to actively contribute to proactive accident avoidance. This aspect not only addresses the immediate safety concerns but also aligns with the broader goal of promoting responsible driving practices in challenging terrains.

Overall, the project's need arises from the critical safety issues prevalent in hilly regions, and its scope extends to providing a holistic solution that combines advanced technologies to prevent accidents, detect incidents, and improve overall road safety in these geographically demanding areas.

#### **1.4 DEEP LEARNING:**

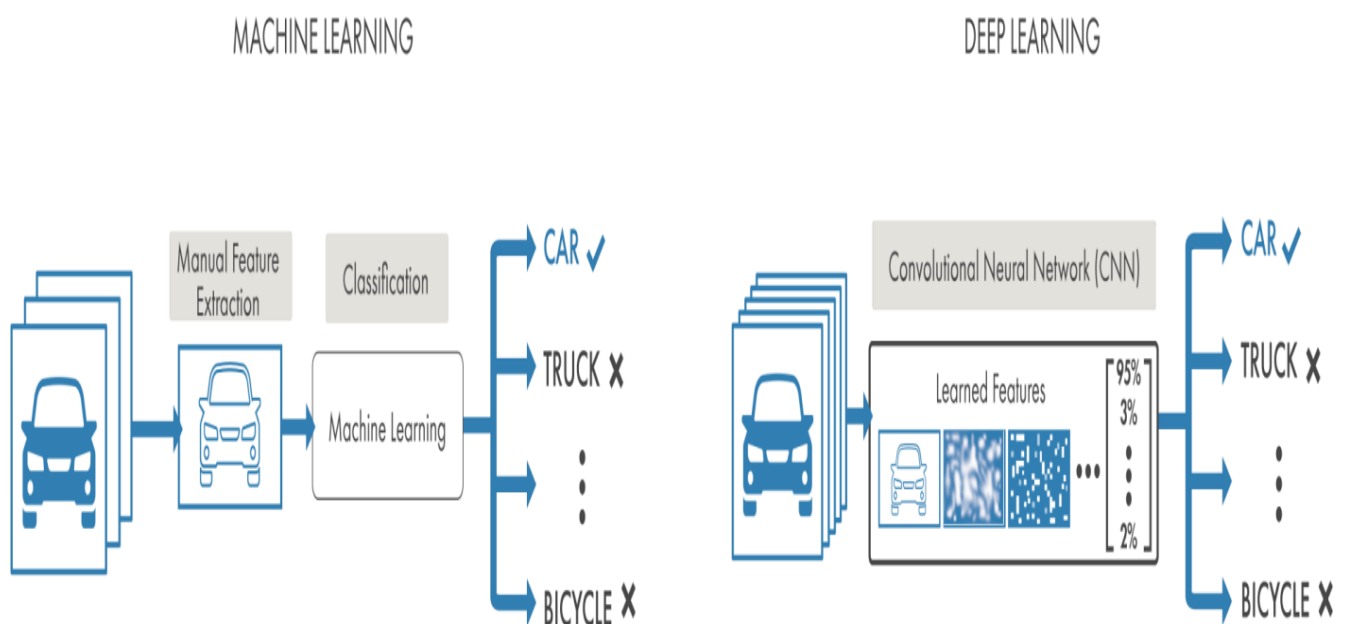
Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain—allowing it to “learn” from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy. In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance.

Models are trained by using a large set of labeled data and neural network architectures that contain many layers. It is a field that is based on learning and improving on its own by examining computer algorithms.

However, advancements in Big Data analytics have permitted larger, sophisticated neural networks, allowing computers to observe, learn, and react to complex situations faster than humans.

#### 1.4.1 DEEP LEARNING vs MACHINE LEARNING:

Deep learning is a specialized form of machine learning. A machine learning workflow starts with relevant features being manually extracted from images. The features are then used to create a model that categorizes the objects in the image. With a deep learning workflow, relevant features are automatically extracted from images. In addition, deep learning performs “end-to-end learning” where a network is given raw data and a task to perform, such as classification, and it learns how to do this automatically. Another key difference is deep learning algorithms scale with data, whereas shallow learning converges. A key advantage of deep learning networks is that they often continue to improve as the size of your data increases. That is, in machine learning, a programmer must intervene directly in the action for the model to come to a conclusion. In the case of a deep learning model, the feature extraction step is completely unnecessary.



**Fig.1.1 Machine learning vs Deep learning**

The model would recognize these unique characteristics of a car and make correct predictions. Deep Learning models tend to increase their accuracy with the increasing amount of training data, where's traditional machine learning models such as SVM and Naive Bayes classifier stop improving after a saturation point

## **CHAPTER 2**

### **LITERATURE SURVEY**

#### **2.1 SURVEY OF THE RELATED WORKS:**

##### **2.1.1 An IOT Based Smart System for Accident Prevention and Detection**

**Author:** Sayanee Nanda; Harshada Joshi; Smita Khairnar

**Year:** 2018

**Description:**

In response to the alarming increase in accidents, particularly bike accidents which constitute a significant portion of overall incidents due to the comparatively fewer safety features in two-wheelers, a comprehensive system is proposed to address this issue. The primary aim of this system is to prevent mishaps and effectively detect and respond to accidents promptly. The system's approach goes beyond simply alerting nearby hospitals, recognizing the potential for secondary accidents. By promptly notifying not only medical facilities but also the person's contacts, it ensures swift response and assistance. Moreover, the system incorporates features to prevent accidents by detecting driver conditions such as drowsiness or instability, which could lead to dangerous scenarios like pedal mix-up or unintended acceleration. This proactive approach aims to address the root causes of accidents by identifying and mitigating risk factors before they escalate.

Furthermore, the system includes a mechanism to verify the rider's eligibility to operate the vehicle by checking for a valid driving license using embedded RFID technology. This additional layer of authentication enhances safety measures by ensuring that only licensed individuals operate the vehicle, thereby reducing the likelihood of accidents caused by inexperienced or unqualified drivers.

### **2.1.2 Intelligent Accident Prevention and Detection for Four -Wheeler**

**Author:** KR. Senthil Murugan; R. Roshni

**Year:** 2021

#### **Description:**

The proposed model for a unique automated accident detection and prevention zone for four-wheelers is a commendable initiative that aims to leverage real-time vehicle tracking (RTVT) technology to enhance emergency response and save lives. The primary objective of this model is to promptly locate accident sites and facilitate the timely dispatch of ambulances, thereby reducing the risk of fatalities and ensuring quicker access to medical care. The significance of this model is highlighted by the common issue of delayed emergency response leading to avoidable casualties. By using GPS and GSM technologies, the system ensures accurate location identification, enabling swift communication with emergency services. The GSM module plays a crucial role in initiating the process of sending the accident location to the relevant authorities, ensuring that crucial information reaches them promptly.

One of the critical aspects addressed by this model is the reduction of traffic to expedite ambulance arrival times at hospitals. The system recognizes the importance of minimizing delays in emergency medical care and aims to improve the overall efficiency of the response process. Instances of lives lost due to the unavailability of ambulances at accident sites and subsequent delays in reaching hospitals underscore the urgency and necessity of such a proactive approach. To further streamline the process, the use of an RF receiver for signal transmission between the ambulance and the system is a practical and innovative addition. The ability to change traffic signals from red to green upon receiving a signal from the ambulance demonstrates an effective mechanism to clear the way and save crucial minutes during emergencies.



### **2.1.3 Accidents Detection and Prevention System to reduce Traffic Hazards using IR Sensors**

**Author:** Naji Taaib Said Al Wadhahi; Shaik Mazhar Hussain

**Year:** 2018

#### **Description:**

The paper addresses the significant global issue of traffic hazards, which pose a major challenge due to the increase in vehicles and dense population, leading to a high number of road accidents and fatalities. Recognizing the urgent need for better transportation facilities to mitigate these hazards and save lives, the paper proposes a solution that utilizes IR sensors and Arduino Uno technology. The proposed solution consists of two main phases: Accident Detection and Accident Prevention. In the Detection phase, IR sensors are employed to detect accidents and promptly alert relevant parties by sending SMS notifications via a GSM module. These notifications include predefined numbers and the accident location obtained from a GPS module, ensuring swift response and assistance.

In the Accident Prevention phase, IR sensors are utilized to warn drivers about neighboring vehicles when the distance between them exceeds a predefined threshold value. This proactive approach aims to prevent accidents by alerting drivers to potential collision risks, thereby enhancing road safety. The paper provides simulation results and presents a prototype to demonstrate the effectiveness of the proposed solution. By leveraging IR sensors and Arduino Uno technology, the solution offers a practical and cost-effective approach to address traffic hazards and improve road safety. Overall, the proposed system holds promise in mitigating traffic hazards and reducing the incidence of road accidents, ultimately contributing to saving lives and enhancing transportation safety. Further research and development in this area could lead to the widespread implementation of similar technologies to benefit communities worldwide

#### **2.1.4 Real time embedded system for accident prevention**

**Author:** Ancy John; P. R. Nishanth

**Year:** 2017

**Description:**

The paper introduces an autonomous accident prevention system with security features, speed control, and accident detection capabilities. The primary goal is to develop an Atmega328P controller that can effectively monitor specific zones using an embedded system, automatically identify the location of an accident, and alert relevant individuals. This is crucial as individuals involved in accidents may not be able to manually convey information. The proposed system comprises two distinct units: the transmitter unit and the receiver unit. As a vehicle approaches the transmitter zone, the system controls its speed by receiving signals from the RF transmitter, strategically placed a few meters before the designated zone. This speed control mechanism aims to enhance safety and reduce the likelihood of accidents.

The security aspect of the system is reinforced by incorporating alcohol, eye, and smoke sensors. These sensors contribute to ensuring that the driver is in a fit state to operate the vehicle, adding an extra layer of accident prevention. The accident detection system is a key component of the proposed solution. In the event of a collision, a piezoelectric sensor detects the impact and relays the signal to the ATmega328P microcontroller. Subsequently, the GPS module in the smartphone communicates with satellites to acquire latitude and longitude values, sending the accident location details to pre-set phone numbers, including those of relatives and emergency services. The integration of GPS and GSM technologies allows for real-time communication and swift response, potentially reducing the emergency response time and improving outcomes for individuals involved in accidents.

### **2.1.5 Accident Prevention System using Machine Learning**

**Author:** Srihan Thokala; Rohith Jakkani; Murari Alli

**Year:** 2017

#### **Description:**

The presented project addresses a critical issue in road safety, emphasizing the high occurrence of accidents during nighttime due to improper visibility. To mitigate this problem, the project proposes a Machine Learning model designed to continuously monitor the road within a 20-meter range, specifically identifying the presence of people or animals crossing the road.

The significance of this project lies in its potential to prevent accidents and save lives, particularly during low-light conditions where human vision is compromised. By employing a Machine Learning model, the system aims to provide an automated and accurate detection mechanism, contributing to the safety of both pedestrians and animals. The primary purpose of the project is to detect pedestrians, including strollers, and animals like dogs during the night and in dim light conditions. This is a crucial aspect, as reduced light intensity poses challenges for human eyes to detect individuals or obstacles on the road. The project acknowledges the limitations of existing systems, particularly their lower accuracy, and seeks to overcome these challenges by implementing a more precise algorithm.

The night-vision system proposed in this project operates on image processing, utilizing a camera and processing units. By leveraging Machine Learning, the system aims to enhance accuracy and effectiveness in detecting potential hazards on the road, thus minimizing the risk of accidents. The project not only addresses the common cause of accidents during nighttime but also emphasizes the importance of leveraging advanced technologies, particularly Machine Learning and image processing, to improve the accuracy and efficiency of existing detection systems.

## 2.2 SUMMARY OF LITERATURE REVIEW

S.NO	Title of the paper	Description
1.	An IOT Based Smart System for Accident Prevention and Detection	In response to the alarming increase in accidents, particularly bike accidents which constitute a significant portion of overall incidents due to the comparatively fewer safety features in two-wheelers, a comprehensive system is proposed to address this issue. The primary aim of this system is to prevent mishaps and effectively detect and respond to accidents promptly. To achieve this, the system utilizes vibration sensors and accelerometers for accident detection, along with GPS and GSM modules to pinpoint the accident location and notify relevant parties such as emergency services and the individual's contacts.
2.	Intelligent Accident Prevention and Detection for Four -Wheeler	The proposed model for a unique automated accident detection and prevention zone for four-wheelers is a commendable initiative that aims to leverage real-time vehicle tracking (RTVT) technology to enhance emergency response and save lives. The primary objective of this model is to promptly locate accident sites and facilitate the timely dispatch of ambulances, thereby reducing the risk of fatalities and ensuring quicker access to medical care. The significance of this model is highlighted by the common issue of delayed emergency response leading to avoidable casualties. By using GPS and GSM technologies, the system ensures

		accurate location identification, enabling swift communication with emergency services. The GSM module plays a crucial role in initiating the process of sending the accident location to the relevant authorities, ensuring that crucial information reaches them promptly.
3.	Accidents Detection and Prevention System to reduce Traffic Hazards using IR Sensors	The paper addresses the significant global issue of traffic hazards, which pose a major challenge due to the increase in vehicles and dense population, leading to a high number of road accidents and fatalities. Recognizing the urgent need for better transportation facilities to mitigate these hazards and save lives, the paper proposes a solution that utilizes IR sensors and Arduino Uno technology. The proposed solution consists of two main phases: Accident Detection and Accident Prevention.
4.	Real time embedded system for accident prevention	The paper introduces an autonomous accident prevention system with security features, speed control, and accident detection capabilities. The primary goal is to develop an Atmega328P controller that can effectively monitor specific zones using an embedded system, automatically identify the location of an accident, and alert relevant individuals. This is crucial as individuals involved in accidents may not be able to manually convey information. The proposed system comprises two distinct units: the transmitter unit and the receiver unit.

5.	Accident Prevention System using Machine Learning	<p>The presented project addresses a critical issue in road safety, emphasizing the high occurrence of accidents during nighttime due to improper visibility. To mitigate this problem, the project proposes a Machine Learning model designed to continuously monitor the road within a 20-meter range, specifically identifying the presence of people or animals crossing the road.</p> <p>The significance of this project lies in its potential to prevent accidents and save lives, particularly during low-light conditions where human vision is compromised.</p>
----	---	---

## **CHAPTER 3**

### **EXISTING SYSTEM**

#### **3.1 EXISTING WORK:**

In hilly areas, the current systems for road safety predominantly rely on traditional measures such as warning signs, speed limits, and occasional patrols. While these measures are essential, they often fall short in providing real-time and proactive accident detection capabilities. The challenging terrain of hilly regions, characterized by sharp turns and steep gradients, demands a more sophisticated approach to ensure road safety. Basic surveillance cameras may be present, but their utility is limited, and they may not cover the entire road network. As a result, there is a critical need to augment these traditional methods with advanced technologies like sensors, the Internet of Things (IoT), and artificial intelligence (AI) to create a more comprehensive and responsive safety infrastructure. Implementing a holistic solution that integrates sensors, IoT devices, and AI algorithms can revolutionize road safety in hilly areas. Sensors strategically placed along roads can monitor various parameters such as road conditions, weather, and vehicular movement. These sensors can be interconnected through an IoT network, enabling real-time data collection and transmission. AI algorithms can then analyze this data to detect potential hazards, predict risky situations, and trigger immediate alerts or warnings. This proactive approach allows for quicker response times, enabling authorities to address emerging issues promptly. For instance, if a landslide is detected or adverse weather conditions are anticipated, automated warnings can be disseminated to drivers, and necessary precautions can be taken to prevent accidents.

The incorporation of advanced technologies not only improves accident detection but also contributes to overall safety by enabling predictive analysis and smarter decision-making. The synergy between sensors, IoT, and AI provides a more robust and adaptive system that can evolve with changing conditions, making road travel in hilly areas safer for both commuters and local residents.

Once a vehicle is detected on the road, the system uploads the relevant data to **Google Firebase**, a secure real-time database service that enables instant data storage and retrieval. Simultaneously, a **NodeMCU microcontroller** retrieves the stored information and activates a **signal pole** (red light) to alert vehicles on the other side of the road about oncoming traffic. This ensures early warnings for drivers, allowing them to proceed cautiously in areas with limited visibility, reducing the probability of collisions. The entire system operates on an **IoT framework** with advanced **end-to-end encryption** to maintain data integrity and prevent any tampering. The use of Firebase ensures seamless communication between system components, providing rapid data processing and responsive alerts. The integration of IoT enables **real-time monitoring** and facilitates seamless data transfer, making the system reliable, efficient, and scalable for real-world deployment in challenging terrains.

In conclusion, the proposed system combines **YOLO for object detection, IoT for communication, and Firebase for real-time data management** to enhance road safety in mountainous regions.

By offering timely alerts and monitoring traffic activity, this system has the potential to **prevent accidents, save lives**, and provide a safer driving environment in areas prone to road mishaps.

### **3.2 PROBLEM FACED IN EXISTING SYSTEM:**

In hill regions, the distinctive topography creates a set of challenges for effective accident detection and prevention. The presence of steep slopes, sharp curves, and unpredictable weather conditions significantly heightens the risks associated with road travel in these areas. Existing solutions, often relying on traditional measures, may prove insufficient to address the complex and dynamic nature of these challenges. Hence, there is a pressing need to develop a specialized system that can seamlessly integrate advanced sensors, real-time weather monitoring, and cutting-edge AI algorithms.



To overcome the limitations of existing solutions, a tailored system for hilly terrains must leverage advanced technologies. Implementing state-of-the-art sensors capable of detecting variations in road conditions, vehicle movements, and environmental factors is crucial. Additionally, incorporating real-time weather monitoring ensures that the system can adapt to rapidly changing conditions, such as sudden fog or heavy rainfall, which are common in hilly regions.

The integration of AI algorithms further enhances the system's capabilities, enabling it to analyze complex data patterns and make accurate predictions about potential accident risks. The ultimate goal of developing such a comprehensive system is to significantly improve the safety of both drivers and passengers navigating the challenging roads in hilly terrains. By proactively identifying potential hazards and providing timely alerts or interventions, this tailored solution aims to reduce the frequency and severity of accidents in these regions. Embracing technological advancements in this manner ensures that safety measures align with the unique challenges presented by the topography of hill areas, thereby creating a more resilient and adaptive approach to accident detection and prevention.

## CHAPTER 4

### PROPOSED SYSTEM

#### 4.1 INTRODUCTION:

The proposed system introduces an innovative solution to address the persistent issue of road safety in mountainous regions, where challenging terrains, sharp curves, and limited visibility often lead to accidents. These conditions make it difficult for drivers to anticipate oncoming traffic, increasing the likelihood of collisions. To overcome these challenges, the system emulates the conditions of mountainous areas using a **comprehensive prototype** designed to monitor and alert vehicles in real time. By leveraging modern technologies such as **Computer Vision**, **Internet of Things (IoT)**, and **real-time data processing**, the system provides a proactive approach to preventing road mishaps. At the core of the system are **two cameras** strategically positioned to capture real-time video feeds of the road. These feeds are processed every second and serve as input for the **Vehicle Detection System (VDS)**, which uses **YOLO (You Only Look Once)**, a state-of-the-art object detection algorithm. YOLO efficiently detects and classifies objects, identifying vehicles (toy cars in the prototype) with high accuracy and speed. Once a vehicle is detected, the system compresses and processes the data, preparing it for further action. This ensures the system operates efficiently even under real-time constraints.

The processed data is then uploaded to **Google Firebase**, a real-time database that serves as the central repository for vehicle detection data. Firebase not only ensures fast and secure data storage but also allows seamless data retrieval when required. To alert drivers about detected vehicles, a **NodeMCU microcontroller** fetches the data from Firebase and activates a **signal pole** equipped with a red light. The red light serves as a visual warning to oncoming traffic, indicating the presence of a vehicle on the other side of the curve or blind spot. This simple yet effective signaling mechanism enhances driver awareness, reducing the likelihood of accidents.

A key strength of the system is its use of **IoT communication with end-to-end encryption**, which ensures the security and integrity of transmitted data. By incorporating robust encryption protocols, the system prevents unauthorized access and data tampering, making it a reliable solution for real-world implementation. The use of Google Firebase further complements the IoT framework, enabling efficient, real-time data management while maintaining a high level of system responsiveness. In summary, the proposed system offers a **holistic solution** to road safety in mountainous regions by combining advanced technologies such as **YOLO for object detection, Firebase for real-time data storage, and NodeMCU for signaling mechanisms**.

#### **4.2 PROPOSED WORK:**

The proposed system introduces a comprehensive prototype designed to emulate the conditions of a mountainous region, addressing the unique challenges associated with road safety in such environments. The system incorporates two cameras that capture real-time video feeds of the road, feeding these images into the system every second. To identify vehicles on the road, the system employs YOLO (You Only Look Once) object detection, a state-of-the-art algorithm capable of efficiently detecting and classifying objects, in this case, toy cars. Once a vehicle is detected, the system processes and compresses the data before uploading it to a Firebase server. Firebase serves as the central repository for storing and managing the vehicle detection data. Simultaneously, a NodeMCU microcontroller fetches this data from Firebase and triggers a signal on a pole, indicating the presence of a vehicle on the other side. This signaling mechanism utilizes a red light to communicate the potential risk to oncoming traffic.

The proposed system leverages IoT with advanced end-to-end encryption, ensuring the integrity and security of the data transmission. This robust network protocol safeguards against tampering, enhancing the overall reliability and trustworthiness of the system.

The choice of Google Firebase's real-time database service further facilitates seamless integration with the system, allowing efficient uploading and retrieval of data based on real-time requirements. In summary, the proposed system offers an innovative solution to enhance safety in mountainous regions by combining advanced object detection technology, secure IoT communication, and real-time data storage. By employing YOLO for vehicle detection, Firebase for data management, and NodeMCU for signaling, the system presents a holistic approach to address the challenges of road safety in challenging terrains.

Here the proposed work is divided into two modules. They are,

- Object detection module
- Node module

In the proposed system, two key modules play pivotal roles in ensuring effective object detection and seamless integration with IoT.

### **OBJECT DETECTION MODULE USING YOLO:**

- The object detection module utilizes the YOLO (You Only Look Once) model for efficient and real-time object detection. YOLO is renowned for its ability to rapidly and accurately identify objects within an image or video frame.
- In this specific case, the YOLO model is applied to detect vehicles on the road. As images from two cameras continuously feed into the system, the YOLO object detection algorithm processes and identifies the presence of objects, particularly toy cars, in the captured frames.
- The successful implementation of the YOLO model ensures robust and timely detection of vehicles, a critical component in addressing safety concerns in mountainous regions.

## **NODE MODULE FOR IOT INTEGRATION:**

- The Node module serves as the IoT component responsible for integrating the system with the physical world. A NodeMCU microcontroller is likely utilized for this purpose.
- Once the object detection module identifies a vehicle on the road and uploads this information to the Firebase server, the Node module comes into play.
- The NodeMCU fetches the relevant data from the Firebase server and translates it into a tangible signal. In this context, the signal is used to control a red light on a pole, serving as a visual indicator for oncoming traffic about the presence of a vehicle on the other side.
- The Node module essentially bridges the digital and physical realms, providing a practical means of communication and alerting based on the information processed by the object detection module.

Together, these modules create a synergistic system where YOLO efficiently detects objects, and the Node module translates this information into actionable signals, enhancing safety measures in mountainous regions.

## **4.2 PROPOSED SYSTEM METHODOLOGY**

The proposed system methodology outlines the step-by-step process and approach that will be followed to implement the system. Below is a generalized framework for the methodology:

### **System Design and Requirement Analysis:**

- Define the objectives and goals of the proposed system, emphasizing the need for advanced object detection and IoT integration in a mountainous region.
- Conduct a thorough analysis of system requirements, considering factors such as topography, lighting conditions, and potential risks.

## **PROTOTYPE DEVELOPMENT:**

- Develop a physical prototype that emulates the environmental conditions of a mountainous region. This may include setting up two cameras for video feeds, integrating a YOLO model for object detection, and implementing a Node module for IoT connectivity.

## **OBJECT DETECTION USING YOLO:**

- Integrate the YOLO object detection model into the system for real-time identification of vehicles, specifically toy cars, from the video feeds.
- Fine-tune the YOLO model parameters to ensure accurate and efficient detection, considering the unique challenges posed by mountainous terrains.

## **DATA PROCESSING AND COMPRESSION:**

- Process the detected object data, including vehicle presence and relevant information, and compress it to optimize storage and transmission efficiency.
- Ensure that the processed data retains essential details while minimizing the overall size for effective communication.

## **FIREBASE INTEGRATION FOR DATA STORAGE:**

- Establish a connection to the Firebase real-time database to store the processed object detection data.
- Implement secure and efficient data transmission protocols to upload and retrieve information from Firebase.

## **Node Module and IoT Integration:**

- Develop the Node module, likely using a NodeMCU microcontroller, to fetch data from Firebase and convert it into a physical signal.
- Implement end-to-end encryption for secure IoT communication, ensuring the reliability and integrity of the signaling process.

### **POLE SIGNALING SYSTEM:**

- Set up a signaling mechanism using a pole-mounted red light to visually indicate the presence of a detected vehicle on the other side of the road.
- Ensure that the signaling system is responsive to real-time updates from the object detection module via the Node module.
- This capability allows it to efficiently process images and detect multiple objects simultaneously, making it a popular choice for various applications, including surveillance, autonomous driving, and environmental monitoring, such as oil spill detection.

### **TESTING AND VALIDATION:**

- Conduct rigorous testing of the entire system under simulated conditions to evaluate the accuracy of object detection, data processing, and signaling mechanisms.
- Validate the system's performance in varying environmental conditions, emphasizing its reliability in mountainous terrains.

### **OPTIMIZATION AND FINE-TUNING:**

- Optimize the system for efficiency, addressing any performance bottlenecks or areas of improvement identified during testing.
- Fine-tune parameters, algorithms, and communication protocols to enhance overall system effectiveness.
- The model utilizes a combination of convolutional layers, attention mechanisms, and enhanced skip connections, allowing it to capture finer details and contextual information from the input images.

## **4.3 SYSTEM REQUIREMENTS:**

### **ALGORITHM USED:**

- YOLO Algorithm

### **HARDWARE USED:**

- Node MCU
- UNO cable
- Esp 8266 - 01 module
- LCD display
- GSM MODULE
- GPS MODULE

### **YOLO V10 ALGORITHM:**

#### **YOLO V10:**

YOLO V10, short for "You Only Look Once Version 8," is an advanced real-time object detection algorithm that builds upon the successes of its predecessors in the YOLO family. Designed for speed and accuracy, YOLO V10 employs a single neural network to predict bounding boxes and class probabilities directly from images in a single evaluation. This capability allows it to efficiently process images and detect multiple objects simultaneously, making it a popular choice for various applications, including surveillance, autonomous driving, and environmental monitoring, such as oil spill detection.

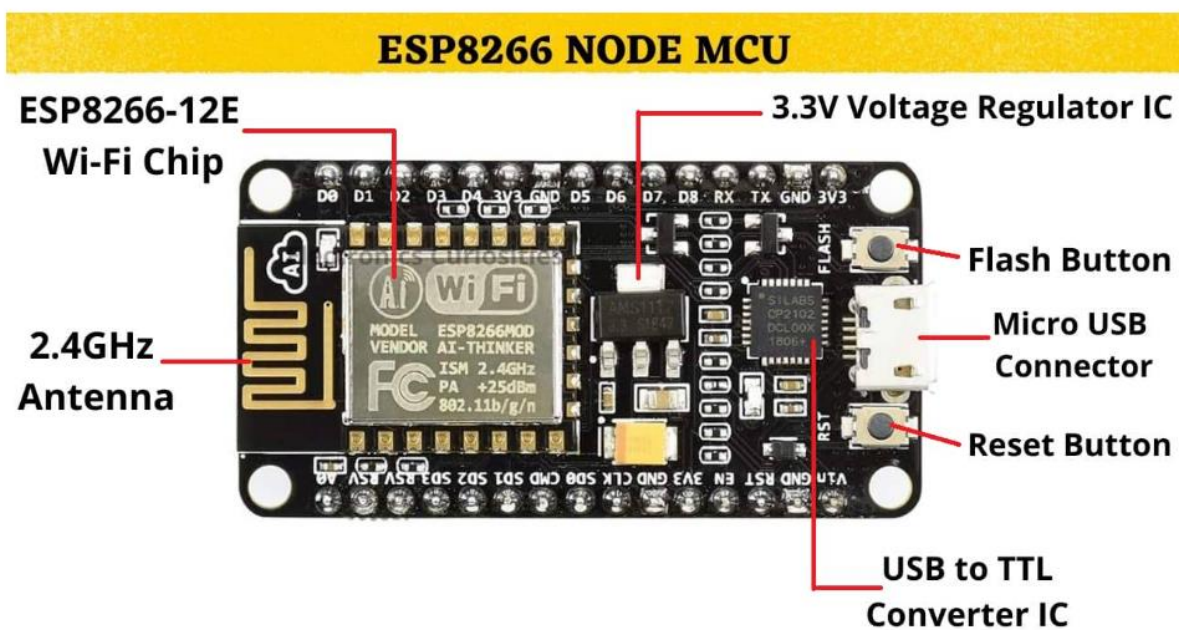
One of the key advancements in YOLO V10 is its improved architecture, which incorporates innovations in backbone networks and feature extraction techniques. The model utilizes a combination of convolutional layers, attention mechanisms, and enhanced skip connections, allowing it to capture finer details and contextual information from the input images.



These enhancements lead to better performance in detecting small and overlapping objects, which is particularly important in scenarios like oil spill detection, where spills may be partially obscured or blend with surrounding water.

Performance analysis of the YOLO (You Only Look Once) algorithm involves assessing its accuracy and speed in object detection tasks. The primary metrics used for performance evaluation are precision, recall, F1 score, and frames per second (FPS).

## **NODE MCU:**

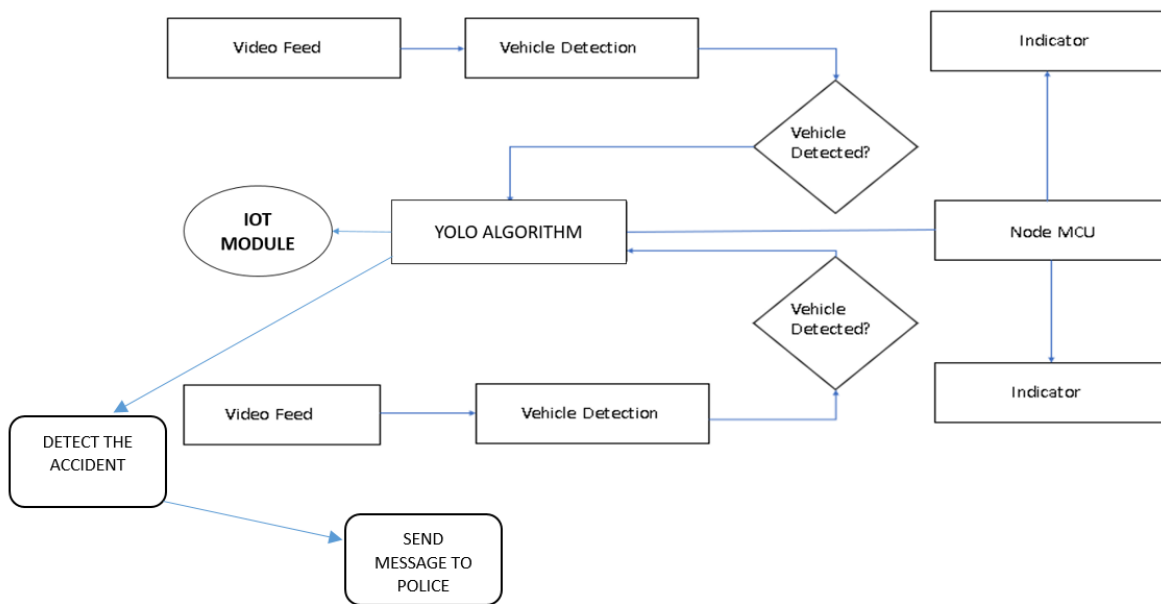


**Fig. 1.2 Node MCU Chipset**

The NodeMCU is a versatile open-source development board based on the ESP8266 Wi-Fi module. Its primary function is to provide a platform for IoT (Internet of Things) projects, enabling easy connectivity and communication with the internet. The NodeMCU integrates a microcontroller unit with built-in Wi-Fi capabilities, making it well-suited for applications that require wireless connectivity. It utilizes the Lua scripting language for programming, simplifying the development process for users with varying levels of expertise. Equipped with GPIO pins, UART, I2C, and other interfaces, the NodeMCU allows for seamless integration with various sensors, actuators, and communication modules.

This compact and cost-effective board is widely used for IoT applications, home automation, and other projects where wireless connectivity and low power consumption are essential. Its open-source nature and strong community support contribute to its popularity among developers and makers for creating connected and intelligent devices.

#### 4.4 PROPOSED ARCHITECTURE DIAGRAM:



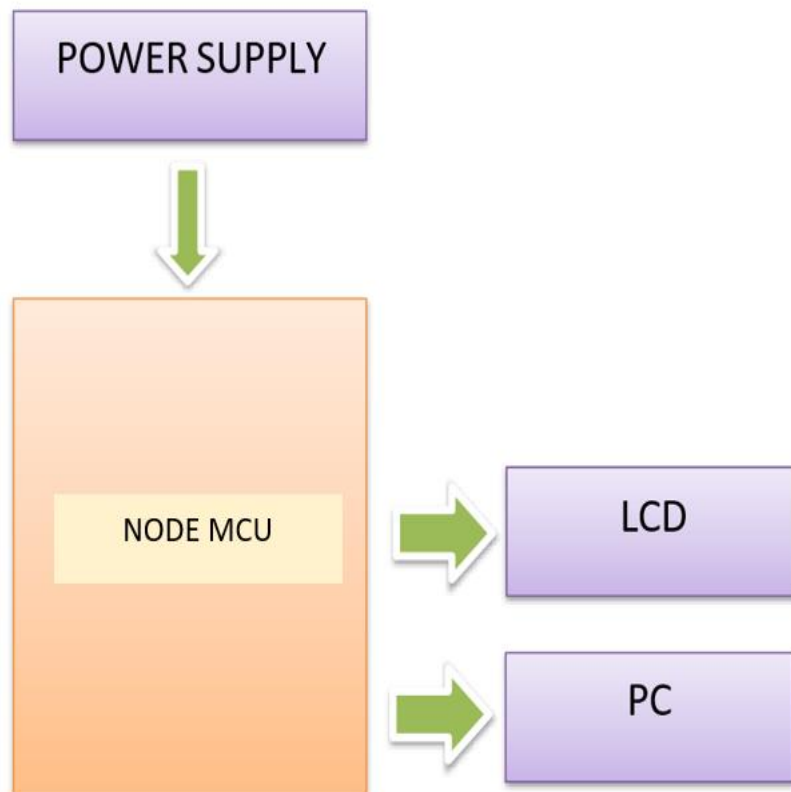
**Fig. 1.3 Architecture Diagram of Proposed Work**

The proposed architecture diagram outlines the systematic design of the envisioned system, aiming to integrate advanced technologies for object detection and IoT communication in a mountainous region. The architecture comprises several key components that collaboratively contribute to the system's functionality. Initially, two cameras capture real-time video feeds of the road environment, feeding the input to the Object Detection Module. This module employs the YOLO (You Only Look Once) model, a powerful object detection algorithm, to identify vehicles, specifically toy cars, in the captured frames. Once detection occurs, the data undergoes processing and compression before being uploaded to a Firebase server, serving as a real-time database

for efficient storage and retrieval. Simultaneously, a Node Module, potentially utilizing a NodeMCU microcontroller, fetches relevant data from Firebase.

The Node Module then translates this information into a tangible signal, specifically a red light on a pole, serving as a visual indicator for oncoming traffic about the presence of a vehicle on the other side of the road. The architecture incorporates end-to-end encryption in the IoT communication process, ensuring secure data transmission. This holistic design aims to enhance safety measures in mountainous terrains, offering a comprehensive solution that combines sophisticated object detection with responsive signaling mechanisms.

#### **4.5 ARCHITECTURE DIAGRAM FOR NODE MCU:**



**Fig. 1.4 Architecture Diagram for Node MCU Chipset**

## **4.6 WORKING:**

The proposed system operates in a systematic manner, leveraging advanced technologies to enhance road safety in mountainous regions. The following is a step-by-step explanation of the working of the system:

### **4.6.1 CAMERA FEEDS AND OBJECT DETECTION:**

- Two cameras continuously capture video feeds of the road environment in the mountainous region.
- The captured video frames are processed using the YOLO (You Only Look Once) object detection model to identify vehicles, specifically toy cars, in real-time.

### **4.6.2 DATA PROCESSING AND COMPRESSION:**

- The detected object data undergoes processing to extract relevant information, such as the presence of a vehicle on the road.
- The processed data is then compressed to optimize storage and facilitate efficient transmission.

### **4.6.3 FIREBASE INTEGRATION FOR DATA STORAGE:**

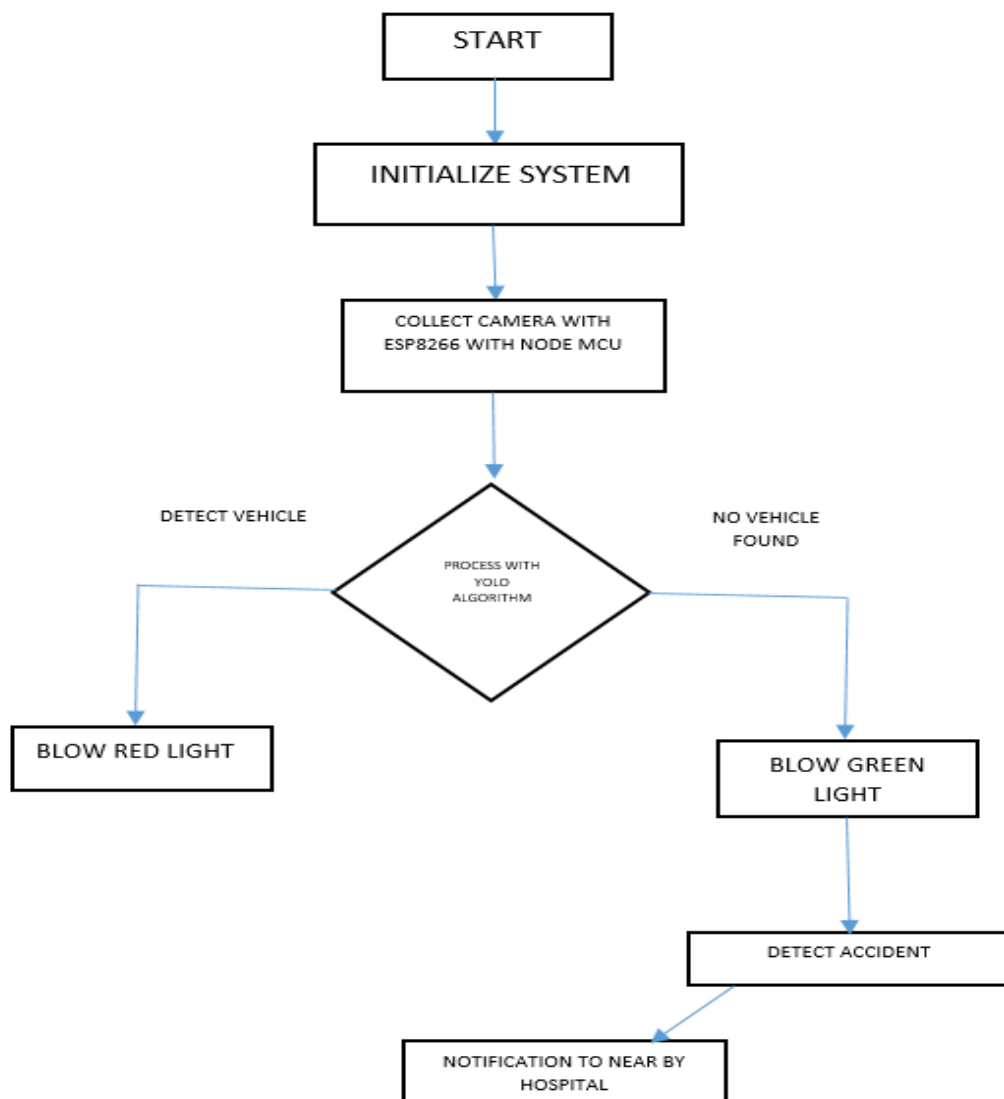
- The compressed data is uploaded to a Firebase server, serving as a real-time database.
- Firebase facilitates secure and seamless storage and retrieval of object detection data.

### **4.6.4 NODE MODULE AND IOT CONNECTIVITY:**

- A Node Module, likely implemented using a NodeMCU microcontroller, fetches the relevant data from Firebase.
- The Node Module serves as the bridge between the digital and physical realms, translating the fetched data into a tangible signal for oncoming traffic.

In summary, the proposed system seamlessly integrates advanced object detection with IoT connectivity, providing a comprehensive solution for proactive road safety in challenging topographies. The continuous monitoring, real-time data processing, and secure communication mechanisms collectively contribute to the overall effectiveness of the system.

#### 4.6.5 WORK-FLOW ARCHITECTURE:



**Fig. 1.5 Work Flow Architecture**

## 4.7 PERFORMANCE ANALYSIS:

Performance analysis of the YOLO (You Only Look Once) algorithm involves assessing its accuracy and speed in object detection tasks. The primary metrics used for performance evaluation are precision, recall, F1 score, and frames per second (FPS).

**Precision (P):** Precision measures the accuracy of positive predictions made by the algorithm. It is calculated using the formula:

$$P = \frac{TP}{TP+FP}$$

where TP is the number of true positives (correctly identified objects) and FP is the number of false positives (incorrectly identified objects).

### **RECALL (R):**

Recall, also known as sensitivity or true positive rate, measures the ability of the algorithm to capture all relevant instances. It is calculated using the formula:

$$R = \frac{TP}{TP+FN}$$

where TP is the number of true positives and FN is the number of false negatives (objects missed by the algorithm).

### **F1 SCORE:**

The F1 score is the harmonic mean of precision and recall, providing a balanced measure of a model's performance. It is calculated using the formula:

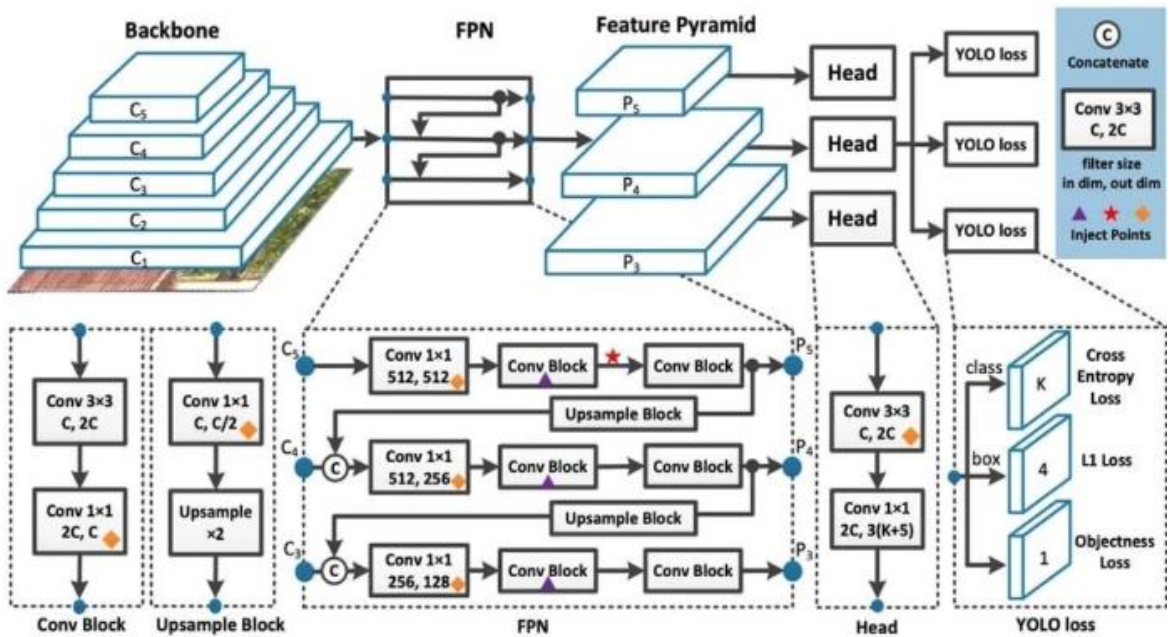
$$F1 = \frac{2 \cdot P \cdot R}{P + R}$$

## FRAMES PER SECOND (FPS):

FPS measures the speed of the algorithm in processing video frames. It represents the number of frames the algorithm can analyze per second. Higher FPS values indicate faster processing speed.

$$FPS = \frac{1}{\text{Average processing time per frame}}$$

Performance analysis involves assessing these metrics on a test dataset, considering the specific object detection task at hand. Precision and recall offer insights into the algorithm's accuracy, while the F1 score provides a balanced evaluation.



In addition to its architectural improvements, YOLO V10 includes advanced training strategies, such as data augmentation, transfer learning, and optimization techniques that help increase the model's robustness and generalization capabilities. By leveraging diverse training datasets and applying various augmentation methods, the model becomes better equipped to handle variations in lighting, weather conditions, and oil spill appearances.

## **CHAPTER – 5**

### **PROPOSED SYSTEM ALGORITHM**

#### **5.1 YOLO V08 ALGORITHM:**

##### **YOLO V08:**

YOLO V08, short for "You Only Look Once Version 8," is an advanced real-time object detection algorithm that builds upon the successes of its predecessors in the YOLO family. Designed for speed and accuracy, YOLO V08 employs a single neural network to predict bounding boxes and class probabilities directly from images in a single evaluation. This capability allows it to efficiently process images and detect multiple objects simultaneously, making it a popular choice for various applications, including surveillance, autonomous driving, and environmental monitoring, such as oil spill detection.

One of the key advancements in YOLO V08 is its improved architecture, which incorporates innovations in backbone networks and feature extraction techniques. The model utilizes a combination of convolutional layers, attention mechanisms, and enhanced skip connections, allowing it to capture finer details and contextual information from the input images.

These enhancements lead to better performance in detecting small and overlapping objects, which is particularly important in scenarios like oil spill detection, where spills may be partially obscured or blend with surrounding water.

Performance analysis of the YOLO (You Only Look Once) algorithm involves assessing its accuracy and speed in object detection tasks. The primary metrics used for performance evaluation are precision, recall, F1 score, and frames per second (FPS).

Object Detection is a computer technology related to computer vision, image processing and deep learning that deals with detecting instances of objects in images and videos. We will do object in this article using something known as haar cascades.



## What are Haar Cascades?

Haar Cascade classifiers are an effective way for object detection. This method was proposed by Paul Viola and Michael Jones in their paper Rapid Object Detection using a Boosted Cascade of Simple Features .Haar Cascade is a machine learning-based approach where a lot of positive and negative images are used to train the classifier.

- **Positive images** – These images contain the images which we want our classifier to identify.
- **Negative Images** – Images of everything else, which do not contain the object we want to detect.

### Requirements:

- Make sure you have python, Matplotlib and OpenCV installed on your pc (all the latest versions).
- The haar cascade files can be downloaded from the OpenCV Github repository.

Object Detection using Haar feature-based cascade classifiers is an effective object detection method. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle. Now all possible sizes and locations of each kernel is used to calculate plenty of features. For each feature calculation, we need to find a sum of pixels under white and black rectangles. To solve this, they introduced the integral images. It simplifies calculation of sum of pixels, how large the number of pixels may be, to an operation involving just four pixels. It makes things super-fast.

## 5.2 PSEUDO CODE:

```
import threading

import requests

import time


app = Flask(__name__, template_folder="templates") # Set template folder


# Load YOLO model

model = YOLO("yolov8n.pt")


HEAVY_VEHICLES = ["bus", "truck"]

LIGHT_VEHICLES = ["car", "motorcycle"]

VEHICLE_LABELS = HEAVY_VEHICLES + LIGHT_VEHICLES


left_detected = False

right_detected = False

lock = threading.Lock()


def send_signal(signal):

    url = f"{ESP_IP}/SIGNAL?data={signal}"

    try:

        response = requests.get(url, timeout=5) # 5 seconds timeout

        if response.status_code == 200:
```

```

def process_frame(frame):

    """Runs YOLO inference, detects vehicles/animals, prioritizes heavy vehicles, and
    sends alerts."""

    global left_detected, right_detected

    height, width, _ = frame.shape

    center_x = width // 2

    detected_left = detected_right = False

    heavy_left = heavy_right = False

    animal_detected = False

    results = model.predict(frame, conf=0.5, verbose=False)

    for r in results:

        for box in r.bboxes:

            label = model.names[int(box.cls[0].item())]

            detected_left = True

            if label in HEAVY_VEHICLES:

                heavy_left = True

            else:

                detected_right = True

```

with lock:

if detected\_left and detected\_right:

if heavy\_left and not heavy\_right:

print("🚚 Heavy vehicle LEFT! Prioritize RIGHT alert.")

send\_signal(1)

left\_detected = True

right\_detected = False

else:

# Default to centroid-based logic if both sides are same type

if not left\_detected:

print("🚗 Vehicles both sides. Defaulting to LEFT priority.")

send\_signal(1)

left\_detected = True

right\_detected = False

elif detected\_left and not left\_detected:

print("🚗 Vehicle LEFT detected. Alert RIGHT.")

send\_signal(1)

left\_detected = True

right\_detected = False

elif detected\_right and not right\_detected:

print("🚗 Vehicle RIGHT detected. Alert LEFT.")

```

send_signal(2)

right_detected = True

left_detected = False


elif not detected_left and not detected_right:

    if left_detected or right_detected:

        print("✔ No vehicle. Clearing alerts.")


    # Process frame with YOLO

    frame = process_frame(frame)

    _, buffer = cv2.imencode('.jpg', frame)

    frame_bytes = buffer.tobytes()

    yield (b'--frame\r\n'
           b'Content-Type: image/jpeg\r\n\r\n' + frame_bytes + b'\r\n')

except Exception as e:

    print(f"Error fetching from camera: {e}")

    continue


@app.route('/video_feed')

def video_feed():

    return Response(generate_frames(), mimetype='multipart/x-mixed-replace;
boundary=frame')
```

This code implements a real-time vehicle detection and signaling system using computer vision and web technologies. It utilizes the YOLOv8 object detection model to identify and classify vehicles from a live video stream captured through an ESP32-CAM. The goal of the system is to monitor traffic, determine the presence and type of vehicles—specifically distinguishing between heavy and light vehicles—and send appropriate control signals to an external ESP-based microcontroller that likely manages some form of traffic response, such as activating signals or alerts.

The application operates by continuously retrieving image frames from a remote camera over a local network. These frames are processed using the YOLOv8 neural network, which detects and classifies objects in the frame. The spatial position of the detected vehicles is analyzed relative to the center of the frame, allowing the system to determine whether vehicles are located on the left or right side. Based on this spatial analysis and the type of vehicle detected, a decision is made about which signal should be sent to the microcontroller.

The Flask web framework is used to create a lightweight web server that serves a front-end HTML page and a video stream endpoint. The video feed shown on the front-end is composed of the YOLO-processed frames, streamed in real-time using MJPEG encoding. This makes the system interactive and suitable for live monitoring applications, allowing users to observe the detection process visually while the backend logic handles decision-making and control signal dispatching.

## CHAPTER – 6

### ALGORITHM AND TECHNOLOGY

#### 6.1 NLP (NATURAL LANGUAGE PROCESSING)

Natural language processing (NLP) is a subfield of computer science and artificial intelligence (AI) that uses machine learning to enable computers to understand and communicate with human language. NLP enables computers and digital devices to recognize, understand and generate text and speech by combining computational linguistics, the rule-based modeling of human language together with statistical modeling, machine learning and deep learning. NLP research has helped enable the era of generative AI, from the communication skills of large language models (LLMs) to the ability of image generation models to understand requests. NLP is already part of everyday life for many, powering search engines, prompting chatbots for customer service with spoken commands, voice-operated GPS systems and question-answering digital assistants on smartphones such as Amazon's Alexa, Apple's Siri and Microsoft's Cortana. NLP also plays a growing role in enterprise solutions that help streamline and automate business operations, increase employee productivity and simplify business processes.

#### 6.2 CNN (CONVOLUTIONAL NEURAL NETWORK)

A Convolutional Neural Network (CNN) is a type of neural network, particularly well-suited for processing data with a grid-like topology, such as images, audio, and time series. CNNs leverage convolution operations to learn features from data, making them efficient for tasks like image recognition, object detection, and classification.

#### 6.3 AI (ARTIFICIAL INTELLIGENCE)

Artificial intelligence is a field of science concerned with building computers and machines that can reason, learn, and act in such a way that would normally require human intelligence or that involves data whose scale exceeds what humans can analyze.

AI is a broad field that encompasses many different disciplines, including computer science, data analytics and statistics, hardware and software engineering, linguistics, neuroscience, and even philosophy and psychology. On an operational level for business use, AI is a set of technologies that are based primarily on machine learning and deep learning, used for data analytics, predictions and forecasting, object categorization, natural language processing, recommendations, intelligent data retrieval, and more.

Artificial intelligence can be organized in several ways, depending on stages of development or actions being performed.

For instance, four stages of AI development are commonly recognized.

- **Reactive machines:** Limited AI that only reacts to different kinds of stimuli based on pre-programmed rules. Does not use memory and thus cannot learn with new data. IBM's Deep Blue that beat chess champion Garry Kasparov in 1997 was an example of a reactive machine.
- **Limited memory:** Most modern AI is considered to be limited memory. It can use memory to improve over time by being trained with new data, typically through an artificial neural network or other training model. Deep learning, a subset of machine learning, is considered limited memory artificial intelligence.
- **Theory of mind:** Theory of mind AI does not currently exist, but research is ongoing into its possibilities. It describes AI that can emulate the human mind and has decision-making capabilities equal to that of a human, including recognizing and remembering emotions and reacting in social situations as a human would.
- **Self-aware:** A step above theory of mind AI, self-aware AI describes a mythical machine that is aware of its own existence and has the intellectual and emotional capabilities of a human. Like theory of mind AI, self-aware AI does not currently exist.



## **CHAPTER 7**

### **MODULE AND MODULE DESCRIPTION**

#### **7.1 MODULE CLASSIFICATION**

The proposed system consist of two modules namely,

- **YOLO MODULE**
- **NODE MCU MODULE**

##### **CAMERA MODULE:**

The Camera Module captures real-time video feeds of the road environment. Multiple cameras may be used to cover different perspectives, providing comprehensive coverage of the area under surveillance.

##### **YOLO (YOU ONLY LOOK ONCE) OBJECT DETECTION MODULE:**

The YOLO module is responsible for object detection in the captured video frames. It efficiently identifies and classifies objects, particularly vehicles such as toy cars, in real-time.

##### **DATA PROCESSING AND COMPRESSION MODULE:**

This module processes the object detection data, extracting relevant information and compressing it to optimize storage and transmission efficiency. It ensures that the data retains essential details while being transmitted and stored effectively.

##### **NODEMCU IOT MODULE:**

The NodeMCU IoT module serves as the bridge between the digital and physical realms. It fetches relevant data from Firebase and translates this information into a tangible signal. It is crucial for integrating the system with the physical world and triggering a signaling mechanism.

### **SIGNALING SYSTEM MODULE:**

The Signaling System Module involves a physical signaling mechanism, such as a red light mounted on a pole. This module visually indicates the presence of a detected vehicle on the other side of the road, providing a clear warning to oncoming traffic.

### **SECURE IOT COMMUNICATION MODULE:**

The Secure IoT Communication Module ensures end-to-end encryption in the communication process between the NodeMCU module and the Firebase server. It enhances the security and reliability of data transmission.

### **FEEDBACK AND OPTIMIZATION MODULE:**

The Feedback and Optimization Module allows for continuous monitoring and feedback, enabling iterative improvements to the system based on performance evaluations and real-world conditions.

These modules work collaboratively to create a comprehensive solution for detecting and preventing accidents in challenging terrains. The integration of advanced object detection with IoT communication and signaling mechanisms ensures a holistic approach to road safety in mountainous regions.

### **NODE MCU:**

NodeMCU is an open-source LUA based firmware developed for the ESP8266 wifi chip. By exploring functionality with the ESP8266 chip, NodeMCU firmware comes with the ESP8266 Development board/kit i.e. NodeMCU Development board. Its applicability to critical areas such as environmental monitoring positions it as a valuable tool for addressing challenges like oil spills, enabling rapid response and effective management strategies



Since NodeMCU is an open-source platform, its hardware design is open for edit/modify/build.

NodeMCU Dev Kit/board consist of ESP8266 wifi enabled chip. The ESP8266 is a low-cost Wi-Fi chip developed by Espressif Systems with TCP/IP protocol. For more information about ESP8266, you can refer to the ESP8266 WiFi Module. There is Version2 (V2) available for NodeMCU Dev Kit i.e. NodeMCU Development Board v1.0 (Version2), which usually comes in black colored PCB.

The NodeMCU is a development board based on the ESP8266 Wi-Fi module, and it typically features a variety of pins, each serving a specific purpose. Below is an explanation of the commonly found pins on a NodeMCU development board:

### **3V3 (3.3V):**

This pin provides a regulated 3.3 volts output. It is used to power external components that require a 3.3V power supply.

### **GND (Ground):**

The ground pin is the reference point for all voltage measurements. It is connected to the ground of the power supply.

**Vin (Voltage In):**

Vin is used to supply an external voltage to the NodeMCU. It can be connected to a power source within the specified voltage range.

**EN (Enable):**

The Enable pin is used to enable or disable the NodeMCU. When pulled high, the module is enabled.

**RST (Reset):**

The Reset pin is used to reset the NodeMCU. Applying a low-level signal to this pin resets the microcontroller.

**D0 to D8 (Digital Pins):**

These pins are general-purpose digital input/output pins. They can be configured as either input or output and are used for interfacing with external sensors, LEDs, or other digital devices.

**A0 (Analog Pin):**

A0 is an analog input pin. It can be used to read analog signals from sensors that provide a variable voltage output.

**TXD and RXD (Transmit and Receive):**

These pins are used for serial communication. TXD is the transmit pin, and RXD is the receive pin.

**SDA and SCL (I2C Pins):**

SDA (Serial Data) and SCL (Serial Clock) are used for I2C communication. These pins are used to connect to devices that support I2C communication.

**D9 and D10 (SPI Pins):**

D9 is the SPI (Serial Peripheral Interface) clock pin, and D10 is the SPI chip select pin. These pins are used for interfacing with devices that communicate using the SPI protocol.

**D5, D6, D7 (PWM Pins):**

These pins support Pulse Width Modulation (PWM) and can be used to control the intensity of LEDs or the speed of motors.

**D1, D2, D3 (Interrupt Pins):**

These pins can be used as interrupt pins to trigger specific actions when a change in state is detected.

Understanding the functions of each pin is crucial when designing circuits or programming the NodeMCU for specific applications. Different projects may use different combinations of these pins based on the requirements of the connected components and sensors.

**7.2 MODULE DESCRIPTION:**

- Video Feed Module
- Vehicle Detection Module
- YOLO Algorithm Module
- IoT Module
- Node MCU Module
- Indicator Module

### **7.2.1 VIDEO FEED MODULE:**

The **Video Feed Module** is the initial input source for the entire system. It consists of cameras placed strategically along the road in mountainous regions to capture real-time video footage. These cameras continuously feed video data into the system, which is essential for detecting objects, such as vehicles, on the road. The primary function of this module is to provide a clear and consistent visual representation of the road conditions in a particular area. The system depends on the quality and consistency of these video feeds to ensure accurate vehicle detection and accident analysis. It serves as the backbone of the system, as the subsequent modules rely on this input to analyze and identify objects of interest, particularly vehicles. This video feed also plays a crucial role in accident detection, as sudden movements or anomalies in the footage can signal the occurrence of a potential accident. By maintaining a continuous flow of high-quality video data, this module ensures that the system can function in real-time, providing immediate responses to traffic changes or emergencies.

### **7.2.2 VEHICLE DETECTION MODULE:**

The **Vehicle Detection Module** is responsible for processing the raw video feed from the cameras. This module identifies and isolates moving objects in the video that may be vehicles. It acts as the first step in filtering relevant data from the video stream before passing it onto the next stage, which involves using the YOLO algorithm for detailed vehicle detection. The vehicle detection process typically involves motion detection and object tracking techniques to identify objects that meet predefined criteria such as size, speed, and movement patterns. This module ensures that the system can efficiently handle large amounts of video data by isolating and tracking moving objects, focusing on vehicles. In the case of a mountainous or hilly terrain, the system is designed to recognize vehicles even in challenging conditions such as limited visibility due to weather or rough terrain. This feature enables the system to make informed decisions about the presence of vehicles on the road and alert other system components accordingly.

### 7.2.3 YOLO ALGORITHM MODULE:

The **YOLO (You Only Look Once) Algorithm** is the core technology used in the system to detect vehicles. YOLO is a powerful real-time object detection algorithm that divides the image into a grid and processes each grid cell to identify objects within it. In this system, the YOLO algorithm is specifically trained to recognize vehicles, which can include cars, trucks, or any other vehicles of interest on the road. YOLO is highly efficient because it makes a single pass over the image to detect multiple objects, unlike traditional object detection algorithms that require multiple passes.

Once the algorithm detects a vehicle, it provides the bounding box coordinates and the associated confidence score indicating the likelihood of the object being a vehicle. This real-time detection capability allows for swift decision-making in time-sensitive scenarios, which is crucial for road safety, particularly in areas prone to accidents or traffic congestion. YOLO's ability to run quickly while maintaining accuracy makes it the ideal choice for this system, where vehicle detection must happen in real-time to alert drivers to potential dangers or obstacles on the road.

### 7.2.4 IOT MODULE:

The **IoT (Internet of Things) Module** plays a crucial role in ensuring that the system components communicate seamlessly. Once the YOLO algorithm detects a vehicle on the road, the IoT module is responsible for transmitting this data to other modules, such as the Node MCU and Indicator modules. The IoT module facilitates the exchange of information between the sensors, the vehicle detection system, and the external devices like traffic lights or warning signals. This communication allows the system to trigger actions, such as illuminating a red light to warn other drivers of the vehicle's presence. The IoT module operates on a network protocol that ensures secure and real-time data transmission. It is essential that this communication is reliable, as it directly affects the effectiveness of the system.

Additionally, the IoT module can be configured to provide cloud integration, allowing the system to upload data to a remote server for long-term analysis or monitoring. The ability to relay information between multiple devices in real-time, and even send alerts or updates to external authorities or users, enhances the overall effectiveness of the system in preventing accidents.

#### **7.2.5 Node MCU Module:**

The **Node MCU Module** is a critical microcontroller within the system that receives information from the IoT module and triggers actions based on that data. In this system, the Node MCU is responsible for activating the visual indicators (such as a red warning light) that alert drivers to the presence of vehicles on the road. Once the IoT module transmits the vehicle detection data, the Node MCU interprets it and makes decisions about how to react based on predefined conditions. For example, when a vehicle is detected on the road, the Node MCU will turn on the red light to signal other drivers of the potential hazard. The module uses simple inputs and outputs to control physical devices, such as lights or alarms, making it a key component in the real-time operation of the system. Additionally, the Node MCU is capable of communication with other parts of the system via Wi-Fi, which ensures that the system is both adaptable and scalable. The use of a Node MCU is beneficial because it is low-cost, efficient, and can be easily integrated into various devices to control physical elements based on real-time data.

#### **7.2.6 INDICATOR MODULE:**

The **Indicator Module** is the visual output mechanism of the system. Once a vehicle is detected by the YOLO algorithm and processed by the IoT and Node MCU modules, the **Indicator Module** is activated to provide warnings to other drivers. This indicator could be in the form of a red light, a traffic sign, or any other visual signal that alerts drivers to the presence of a vehicle on the road, preventing potential collisions.



The indicator serves as a real-time alert to oncoming traffic and is crucial for ensuring safety in hazardous road conditions, especially in mountainous regions where visibility may be poor. The module is connected to the Node MCU, which controls its activation based on the vehicle detection data received. The Indicator Module ensures that drivers are notified promptly, providing them with time to slow down or take precautionary actions to avoid accidents. In this system, the ability to trigger these indicators reliably is essential for maintaining the safety of drivers, particularly in areas with winding roads or challenging weather conditions.

## **CHAPTER 8**

### **PROPOSED SYSTEM ADVANTAGE**

#### **8.1 ADVANTAGE OF THE PROPOSED SYSTEM:**

- The proposed Intelligent Mountainous Region Traffic Monitoring and Alert System offers several significant advantages, particularly in enhancing road safety in challenging terrains.
- One of the key benefits is its ability to provide real-time vehicle detection using the YOLO algorithm, which ensures quick and accurate identification of vehicles on the road.
- This allows the system to promptly alert drivers to potential hazards or vehicles on the opposite side, minimizing the risk of accidents in mountainous areas where visibility can be limited.
- The integration of IoT technology ensures seamless communication between various system components, enabling efficient data transfer and immediate response actions such as activating warning signals.
- Additionally, the use of Firebase for real-time data storage and retrieval further enhances the system's responsiveness, ensuring that alerts and updates are delivered instantly.
- The Node MCU module's ability to trigger indicators based on detected vehicles ensures that physical alerts are provided in real-time, improving driver awareness. Furthermore, the system's advanced encryption protocols ensure data security and integrity, preventing tampering and enhancing trustworthiness.
- Ultimately, the system combines cutting-edge technology to create a reliable, efficient, and secure solution for preventing accidents and improving traffic safety in mountainous regions, potentially saving lives and reducing accidents significantly.

## CHAPTER – 9

### CODING

#### 9.1 APP.PY

```
from flask import Flask, Response, render_template, jsonify
import cv2
import numpy as np
from ultralytics import YOLO
import threading
import requests
import time

app = Flask(__name__, template_folder="templates") # Set template folder

# Load YOLO model
model = YOLO("yolov8n.pt")

# OpenCV video capture
#cap = cv2.VideoCapture(0) # Use default webcam

# Separate heavy vehicle labels
HEAVY_VEHICLES = ["bus", "truck"]
LIGHT_VEHICLES = ["car", "motorcycle"]
VEHICLE_LABELS = HEAVY_VEHICLES + LIGHT_VEHICLES
```

```
CAM_URL = "http://192.168.63.129/cam-mid.jpg"
```

```
ESP_IP = "http://192.168.63.195"
```

```
# Detection flags
```

```
left_detected = False
```

```
right_detected = False
```

```
lock = threading.Lock()
```

```
def send_signal(signal):
```

```
    url = f"{ESP_IP}/SIGNAL?data={signal}"
```

```
    try:
```

```
        response = requests.get(url, timeout=5) # 5 seconds timeout
```

```
        if response.status_code == 200:
```

```
            print(f"✔ Signal {signal} Sent Successfully!")
```

```
            print("Response:", response.text)
```

```
        else:
```

```
            print(f"⚠ Failed to Send Signal {signal}. HTTP Code:", response.status_code)
```

```
            print("Response:", response.text)
```

```
    except requests.exceptions.RequestException as e:
```

```
        print(f"✖ Error Sending Signal {signal}: {e}")
```

```

def process_frame(frame):

    """Runs YOLO inference, detects vehicles/animals, prioritizes heavy vehicles, and
    sends alerts."""

    global left_detected, right_detected

    height, width, _ = frame.shape

    center_x = width // 2

    detected_left = detected_right = False

    heavy_left = heavy_right = False

    animal_detected = False

    results = model.predict(frame, conf=0.5, verbose=False)

    for r in results:

        for box in r.bboxes:

            label = model.names[int(box.cls[0].item())]

            if label in VEHICLE_LABELS:

                x1, y1, x2, y2 = map(int, box.xyxy[0])

                centroid_x = (x1 + x2) // 2

```

```

if centroid_x < center_x:
    detected_left = True
    if label in HEAVY_VEHICLES:
        heavy_left = True
else:
    detected_right = True
    if label in HEAVY_VEHICLES:
        heavy_right = True

```

with lock:

```

if detected_left and detected_right:
    if heavy_left and not heavy_right:
        print("🚚 Heavy vehicle LEFT! Prioritize RIGHT alert.")
        send_signal(1)
        left_detected = True
        right_detected = False
    elif heavy_right and not heavy_left:
        print("🚚 Heavy vehicle RIGHT! Prioritize LEFT alert.")
        send_signal(2)
        right_detected = True
        left_detected = False

```

else:

# Default to centroid-based logic if both sides are same type

if not left\_detected:

print("🚗 Vehicles both sides. Defaulting to LEFT priority.")

send\_signal(1)

left\_detected = True

right\_detected = False

elif detected\_left and not left\_detected:

print("🚗 Vehicle LEFT detected. Alert RIGHT.")

send\_signal(1)

left\_detected = True

right\_detected = False

elif detected\_right and not right\_detected:

print("🚗 Vehicle RIGHT detected. Alert LEFT.")

send\_signal(2)

right\_detected = True

left\_detected = False

elif not detected\_left and not detected\_right:

if left\_detected or right\_detected:

print("✓ No vehicle. Clearing alerts.")

```

        send_signal(0)

    left_detected = right_detected = False

    return frame

def generate_frames():
    """Fetches frames from ESP32-CAM, processes them, and streams via Flask."""
    while True:
        try:
            response = requests.get(CAM_URL, timeout=2)

            if response.status_code == 200:
                img_array = np.frombuffer(response.content, np.uint8)
                frame = cv2.imdecode(img_array, cv2.IMREAD_COLOR)

                # Process frame with YOLO
                frame = process_frame(frame)

                _, buffer = cv2.imencode('.jpg', frame)
                frame_bytes = buffer.tobytes()

                yield (b'--frame\r\n'
                       b'Content-Type: image/jpeg\r\n\r\n' + frame_bytes + b'\r\n')

```



```

except Exception as e:

    print(f"Error fetching from camera: {e}")

    continue


@app.route('/')
def index():

    """Serve the HTML file as the homepage."""

    return render_template("index.html")


@app.route('/video_feed')
def video_feed():

    return Response(generate_frames(), mimetype='multipart/x-mixed-replace;
boundary=frame')


if __name__ == '__main__':

    app.run(host='0.0.0.0', port=5000, debug=True)

```

## **BREAKDOWN:**

### **9.1.1. LIBRARIES & MODEL INITIALIZATION**

#### **Uses:**

- Flask to serve the web interface.
- OpenCV (cv2) for image processing.
- YOLOv8 (from ultralytics) for object detection.
- Loads the **YOLOv8n** model (yolov8n.pt), a lightweight version for faster performance.

### 9.1.2. VEHICLE CLASSIFICATION

- Vehicles are categorized into:
- **Heavy vehicles:** bus, truck
- **Light vehicles:** car, motorcycle

### 9.1.3. ESP32-CAM INTEGRATION

- Video frames are fetched from a URL (<http://192.168.63.129/cam-mid.jpg>) — an image stream from an ESP32-CAM.
- Control signals are sent to another ESP device (<http://192.168.63.195>) with endpoint `/SIGNAL?data=<value>`.

### 9.1.4. FRAME PROCESSING

- The `process_frame()` function:
- Performs object detection on the frame.
- Determines if vehicles are on the **left or right** of the frame center.
- Prioritizes **heavy vehicles** in decision logic.
- Sends control signals:
- 0 – No vehicles, clear signal.
- 1 – Vehicle on **left**, alert to the **right**.
- – Vehicle on **right**, alert to the **left**.

### 9.1.5. STREAMING LOGIC

- `generate_frames()` continuously:
- Grabs an image from the ESP32-CAM.
- Processes the frame with YOLO.
- Streams it as MJPEG to the client via Flask.

### 9.1.6. FLASK ROUTES

- / – Serves index.html from the templates folder.
- /video\_feed – Streams the processed video feed.

### 9.2 ARDUINO. IDE:

```
#define RED1 5
```

```
#define YELLOW1 4
```

```
#define GREEN1 15
```

```
// Traffic Lights 2
```

```
#define RED2 14
```

```
#define YELLOW2 12
```

```
#define GREEN2 13
```

```
// Sensors & Buzzer
```

```
#define TILT_SENSOR 16
```

```
#define BUZZER 2
```

```
// LCD I2C
```

```
#define LCD_SDA 22
```

```

// GPS

#define GPS_TX 17 // RX of ESP32

#define GPS_RX 18 // TX of ESP32


// WiFi & Server Config

const char* ssid = "hairpin1234";

const char* password = "hairpin1234";


IPAddress local_IP(192, 168, 1, 184);

IPAddress gateway(192, 168, 1, 1);

IPAddress subnet(255, 255, 255, 0);


WebServer server(80);


LiquidCrystal_I2C lcd2(0x27, 16, 2); // Single LCD used


int data = 0;


void setup() {

```

```
Serial.begin(115200);
```

```
connectWiFi();
```

```
server.on("/SIGNAL", HTTP_GET, handleUpdate);
```

```
server.begin();
```

```
pinMode(RED1, OUTPUT);
```

```
pinMode(YELLOW1, OUTPUT);
```

```
pinMode(GREEN1, OUTPUT);
```

```
pinMode(RED2, OUTPUT);
```

```
pinMode(YELLOW2, OUTPUT);
```

```
pinMode(GREEN2, OUTPUT);
```

```
pinMode(TILT_SENSOR, INPUT);
```

```
pinMode(BUZZER, OUTPUT);
```

```
Wire.begin(LCD_SDA, LCD_SCL);
```

```
lcd2.begin(16, 2);
```

```
lcd2.backlight();
```

```
updateLCD("Hairpin Safety", "System Ready");
```

```

    controlTraffic(0);

}

void loop() {

    server.handleClient();

    if (data == 1) {

        controlTraffic(1);

    } else if (data == 2) {

        controlTraffic(2);

    }

    else {

        controlTraffic(0);

    }


    delay(500);

}

void connectWiFi() {

    Serial.println("\n🌐 Connecting to WiFi...");

    WiFi.disconnect(true);

```

```

delay(1000);

WiFi.config(local_IP, gateway, subnet);

WiFi.begin(ssid, password);


unsigned long startAttemptTime = millis();


while (WiFi.status() != WL_CONNECTED && millis() - startAttemptTime < 15000)
{

    Serial.print(".");

    delay(1000);

}

if (WiFi.status() == WL_CONNECTED) {

    Serial.println("\n✔ WiFi Connected!");

    Serial.print("📶 ESP32 IP: ");

    Serial.println(WiFi.localIP());

} else {

    Serial.println("\n✗ WiFi Connection Failed! Restarting ESP32...");

    ESP.restart();

}

```

```
}
```

```
void handleUpdate() {
```

```
    if (server.hasArg("data")) {
```

```
        data = server.arg("data").toInt();
```

```
        Serial.println("Received Data: " + String(data));
```

```
        server.send(200, "text/plain", "Data Updated");
```

```
    } else {
```

```
        server.send(400, "text/plain", "Missing 'data' parameter");
```

```
    }
```

```
}
```

```
void updateLCD(String line1, String line2) {
```

```
    lcd2.clear();
```

```
    lcd2.setCursor(0, 0);
```

```
    lcd2.print(line1);
```

```
    lcd2.setCursor(0, 1);
```

```
    lcd2.print(line2);
```

```
}
```



```

void controlTraffic(int side) {

    if (side == 1) {

        digitalWrite(REDD1, LOW);

        digitalWrite(GREEN1, HIGH);

        digitalWrite(YELLOW1, LOW);

        digitalWrite(REDD2, HIGH);

        digitalWrite(GREEN2, LOW);

        digitalWrite(YELLOW2, LOW);

        updateLCD("Lane 1: GO", "Lane 2: STOP");

    } else if (side == 2) {

        digitalWrite(REDD1, HIGH);

        digitalWrite(GREEN1, LOW);

        digitalWrite(YELLOW1, LOW);

        digitalWrite(REDD2, LOW);

        digitalWrite(GREEN2, HIGH);

        digitalWrite(YELLOW2, LOW);

        updateLCD("Lane 1: STOP", "Lane 2: GO");

    }
}

```

```
else {  
  
    digitalWrite(RED1, LOW);  
  
    digitalWrite(GREEN1, LOW);  
  
    digitalWrite(YELLOW1, HIGH);  
  
    digitalWrite(RED2, LOW);  
  
    digitalWrite(GREEN2, LOW);  
  
    digitalWrite(YELLOW2, HIGH);  
  
    updateLCD("No Vehicle", "Proceed with Caution");  
  
}  
  
}
```

## **BREAKDOWN:**

### **9.2.1 Wi-Fi Connection:**

- Connects ESP32 to Wi-Fi with static IP settings.

### **9.2.2 Web Server Setup:**

- Listens for HTTP GET requests at /SIGNAL and accepts a data parameter to control traffic lights.

### **9.2.3 Traffic Control Logic:**

- data == 1: Lane 1 GO, Lane 2 STOP
- data == 2: Lane 2 GO, Lane 1 STOP
- Otherwise: Both lanes show YELLOW ("Proceed with Caution")

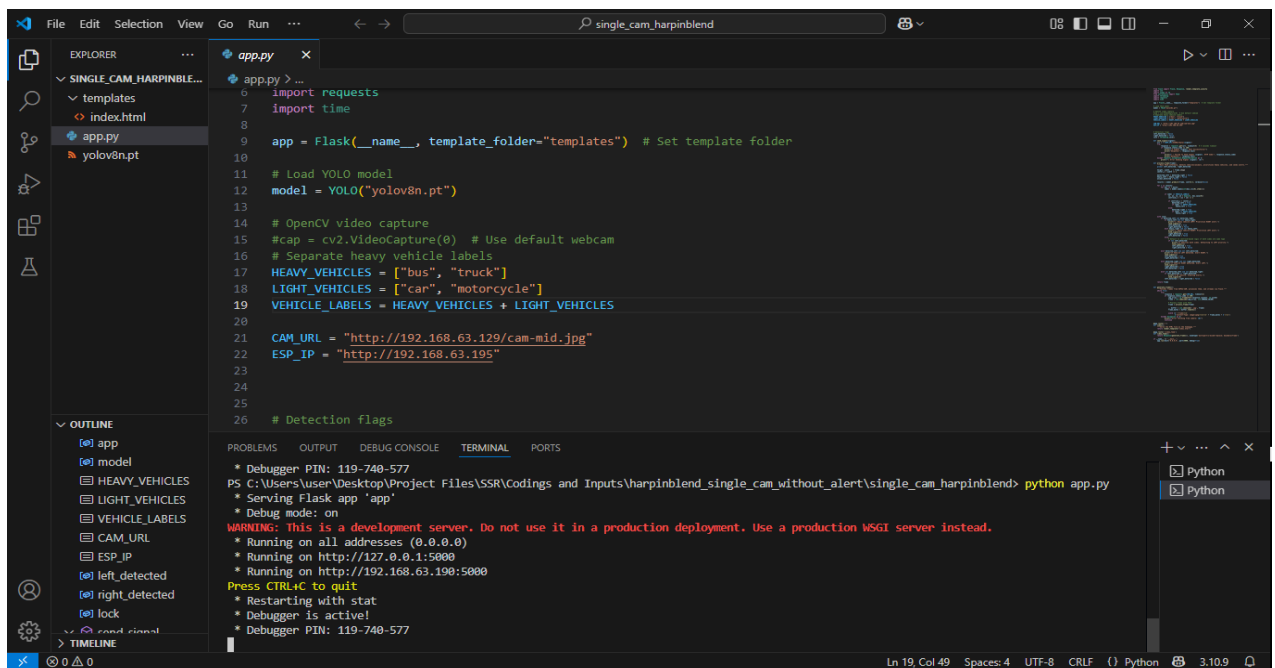
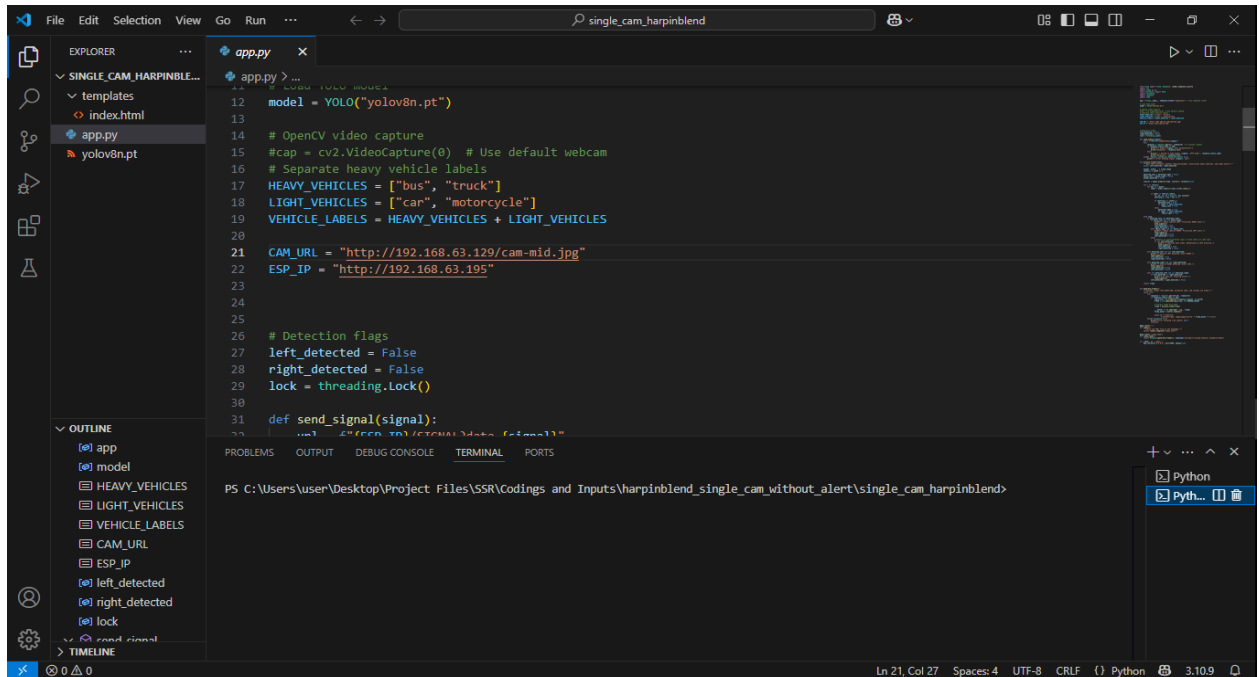
### **9.2.4 LCD Display:**

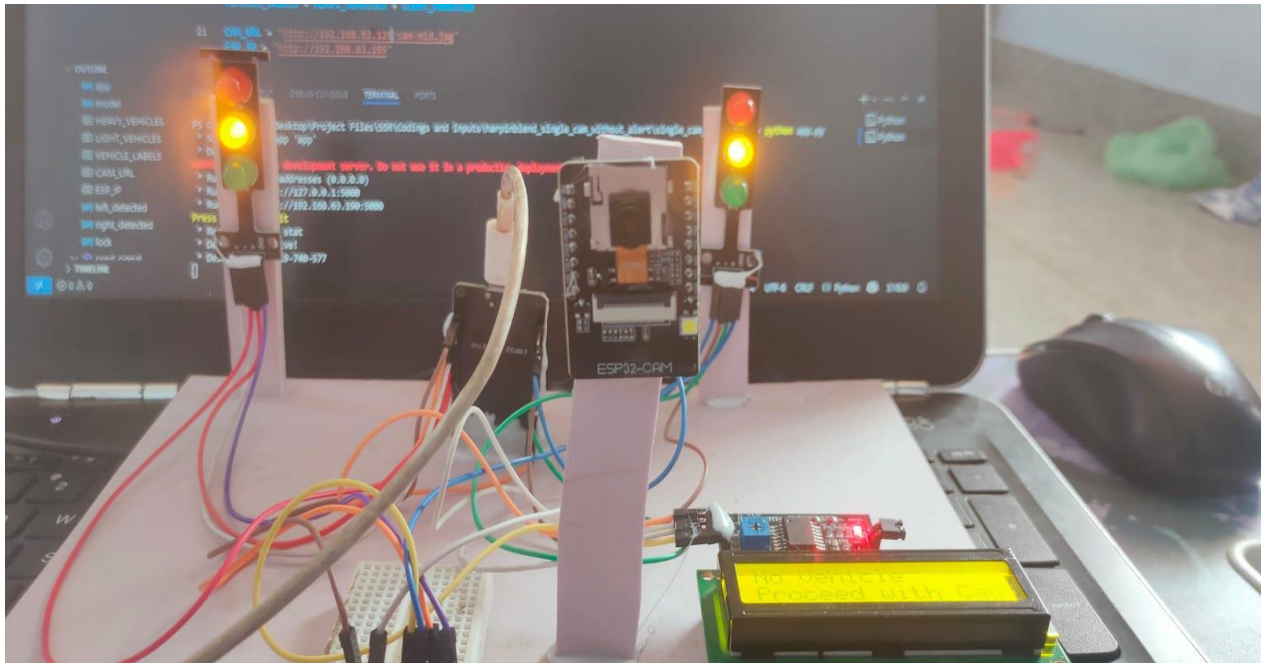
- Provides visual feedback for system readiness and traffic light status. Depending upon the vehicle arrival at the curve. It shows indications to the drivers to make them alert.

## CHAPTER – 10

## SCREENSHOTS

### 10.1 SCREENSHOTS AND OUTPUTS





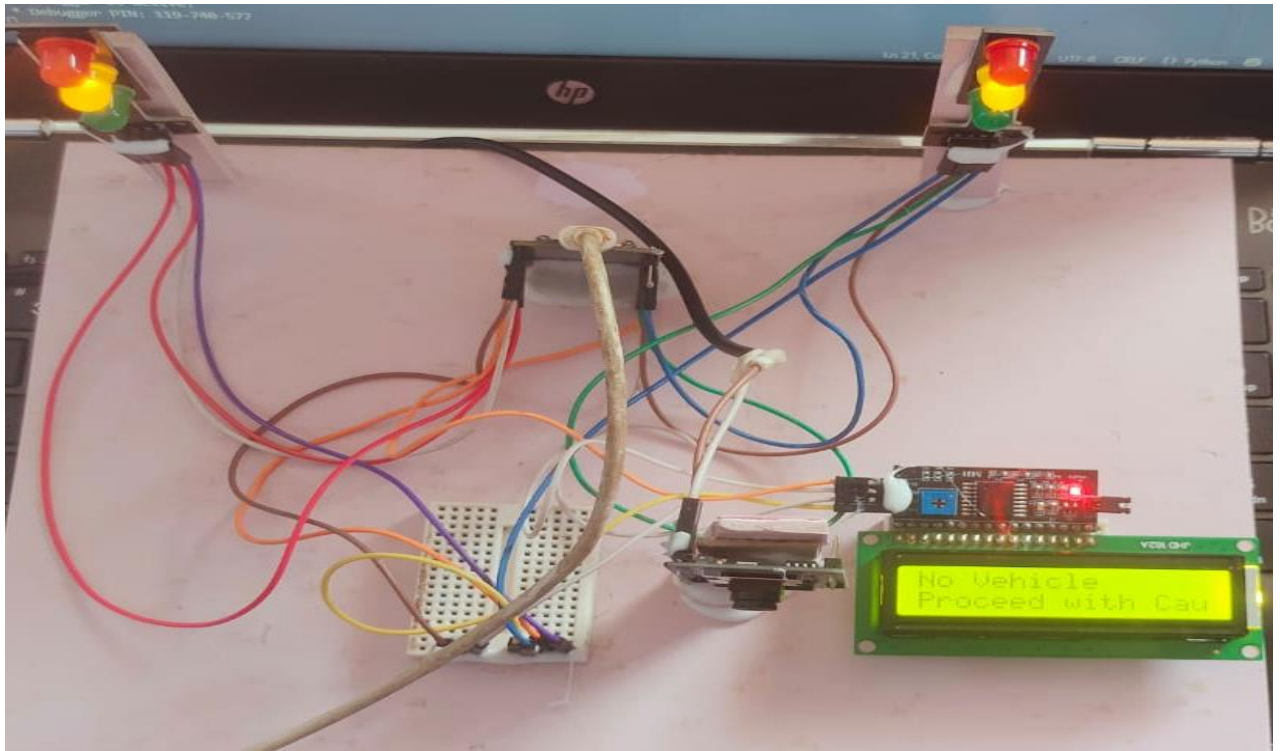
```

File Edit Selection View Go Run ... single_cam_harpinblend
EXPLORER
  SINGLE_CAM_HARPINBLEND
    templates
    index.html
    app.py
    yolov8n.pt
  OUTLINE
    app
    model
    HEAVY_VEHICLES
    LIGHT_VEHICLES
    VEHICLE_LABELS
    CAM_URL
    ESP_IP
    left_detected
    right_detected
    lock
    send_signal
    TIMELINE

app.py
11 # Load YOLO model
12 model = YOLO("yolov8n.pt")
13
14 # OpenCV video capture
15 #cap = cv2.VideoCapture(0) # Use default webcam
16 # Separate heavy vehicle labels
17 HEAVY_VEHICLES = ["bus", "truck"]
18 LIGHT_VEHICLES = ["car", "motorcycle"]
19 VEHICLE_LABELS = HEAVY_VEHICLES + LIGHT_VEHICLES
20
21 CAM_URL = "http://192.168.63.129/cam-mid.jpg"
22 ESP_IP = "http://192.168.63.195"
23
24
25
26 # Detection flags
27 left_detected = False
28 right_detected = False
29 lock = threading.Lock()
30
31 def send_signal(signal):
32     # ... (code for sending signal) ...

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\User\Desktop\Project Files\SSR\Codings and Inputs\harpinblend_single_cam_without_alert\single_cam_harpinblend> python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://192.168.63.190:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 119-740-577

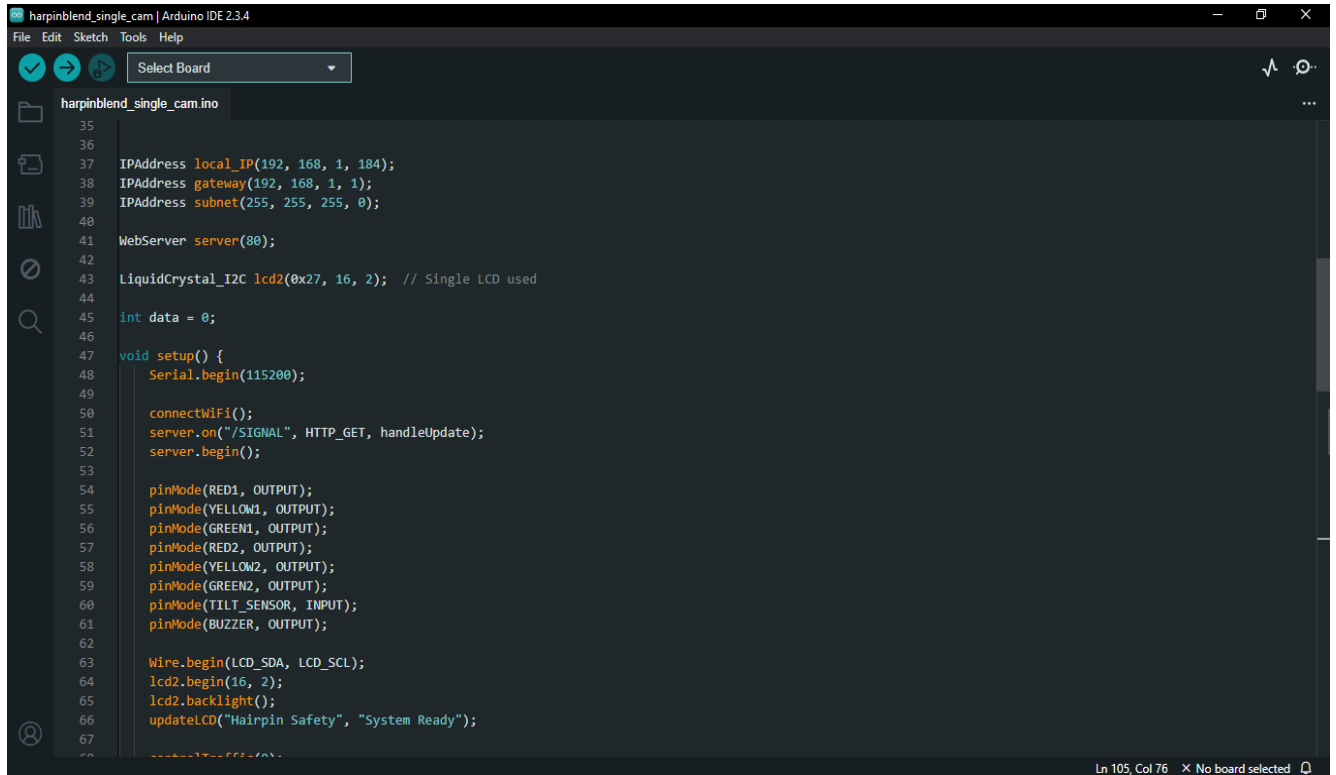
```



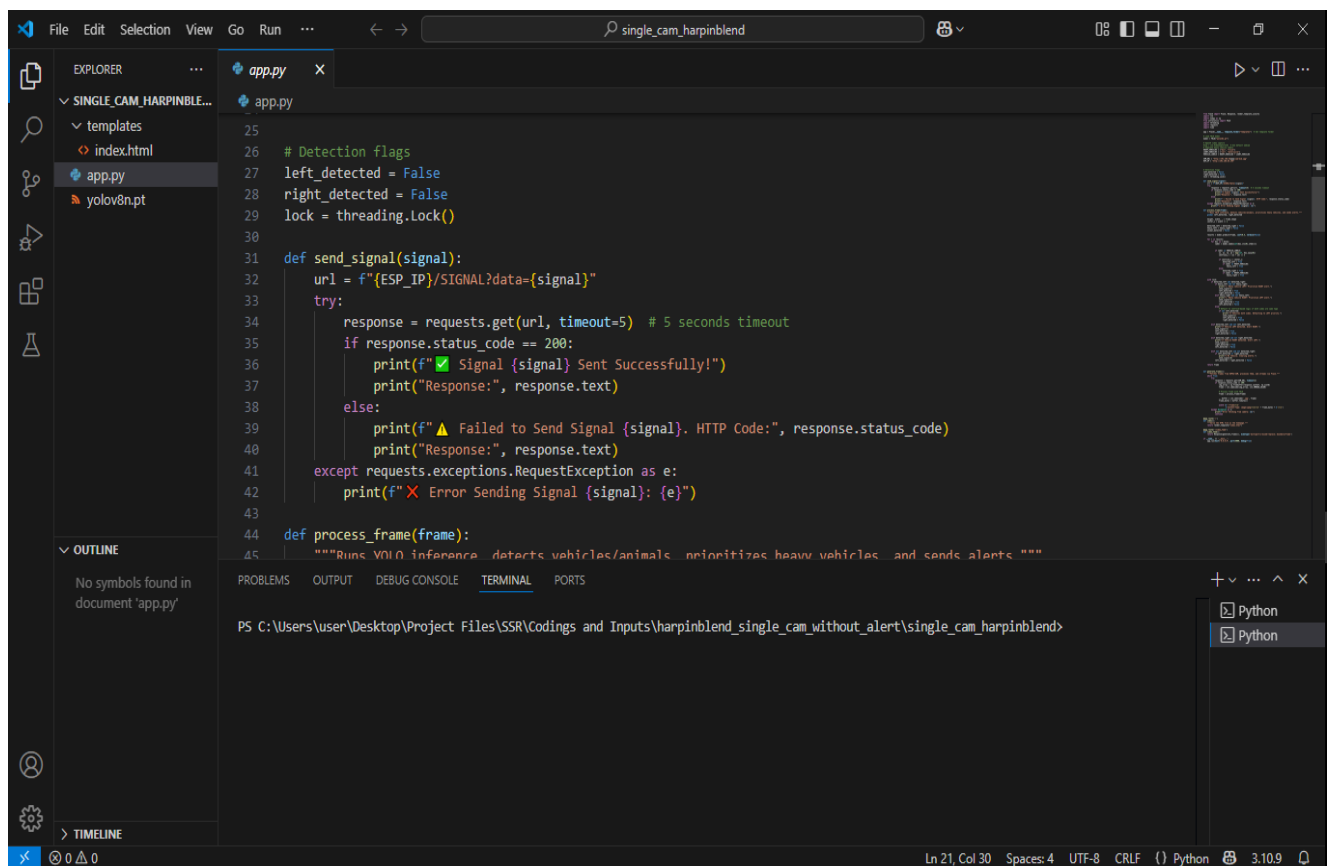
```

File Edit Selection View Go Run ... single_cam_harpinblend
EXPLORER
  SINGL...
  templates
    index.html
  app.py
  yolov8n.pt
OUTLINE
  html
    head
      meta
      meta
      title
    style
      body
        img
      body
        h1
        img
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\user\Desktop\Project Files\SSR\Codings and Inputs\harpinblend_single_cam_without_alert\single_cam_harpinblend> python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://192.168.63.190:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 119-740-577

```

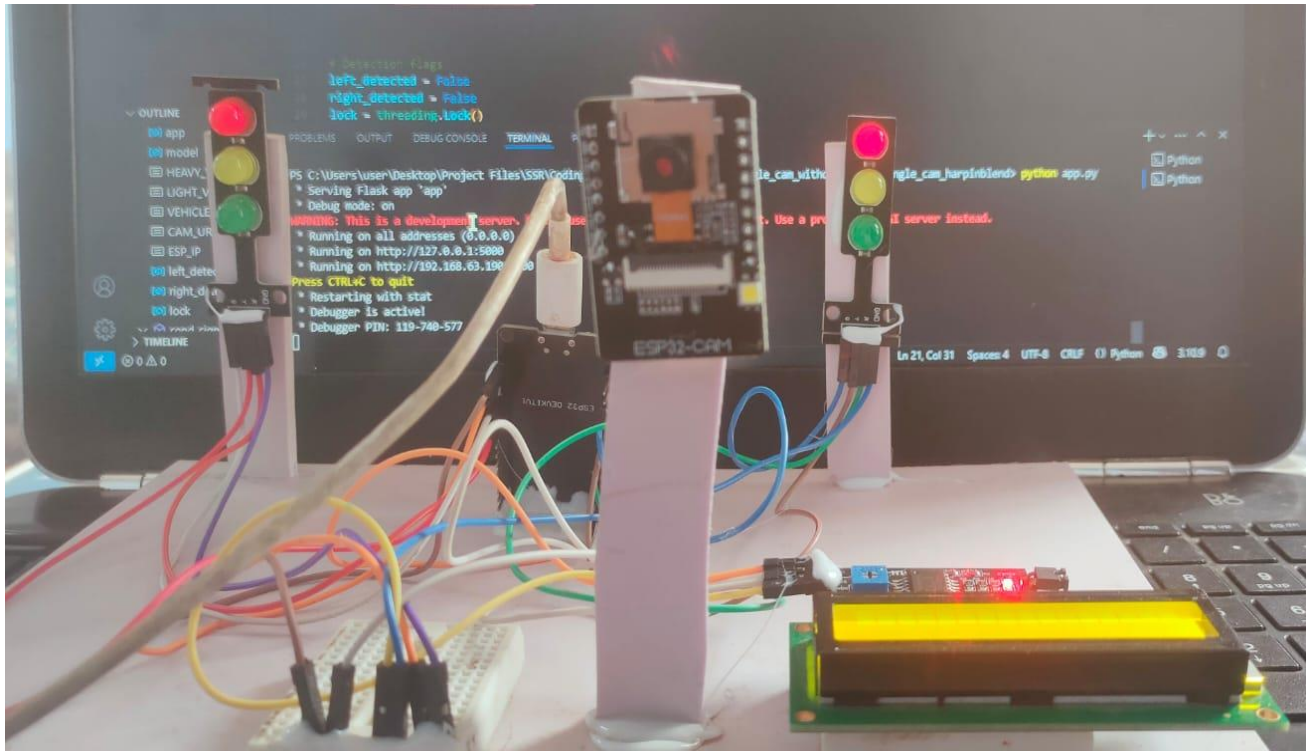


```
35
36
37 IPAddress local_IP(192, 168, 1, 184);
38 IPAddress gateway(192, 168, 1, 1);
39 IPAddress subnet(255, 255, 255, 0);
40
41 WebServer server(80);
42
43 LiquidCrystal_I2C lcd2(0x27, 16, 2); // Single LCD used
44
45 int data = 0;
46
47 void setup() {
48     Serial.begin(115200);
49
50     connectWiFi();
51     server.on("/SIGNAL", HTTP_GET, handleUpdate);
52     server.begin();
53
54     pinMode(REDD1, OUTPUT);
55     pinMode(YELLOW1, OUTPUT);
56     pinMode(GREEN1, OUTPUT);
57     pinMode(REDD2, OUTPUT);
58     pinMode(YELLOW2, OUTPUT);
59     pinMode(GREEN2, OUTPUT);
60     pinMode(TILT_SENSOR, INPUT);
61     pinMode(BUZZER, OUTPUT);
62
63     Wire.begin(LCD_SDA, LCD_SCL);
64     lcd2.begin(16, 2);
65     lcd2.backlight();
66     updateLCD("Hairpin Safety", "System Ready");
67
68     //----->
```



```
25
26 # Detection flags
27 left_detected = False
28 right_detected = False
29 lock = threading.Lock()
30
31 def send_signal(signal):
32     url = f"{ESP_IP}/SIGNAL?data={signal}"
33     try:
34         response = requests.get(url, timeout=5) # 5 seconds timeout
35         if response.status_code == 200:
36             print(f"✅ Signal {signal} Sent Successfully!")
37             print("Response:", response.text)
38         else:
39             print(f"⚠️ Failed to Send Signal {signal}. HTTP Code:", response.status_code)
40             print("Response:", response.text)
41     except requests.exceptions.RequestException as e:
42         print(f"❌ Error Sending Signal {signal}: {e}")
43
44 def process_frame(frame):
45     """Runs YOLOv8 inference, detects vehicles/animals, prioritizes heavy vehicles, and sends alerts """
```

PS C:\Users\user\Desktop\Project Files\SSR\Codings and Inputs\harpinblend\_single\_cam\_without\_alert\single\_cam\_harpinblend>



```

harpinblend_single_cam | Arduino IDE 2.3.4
File Edit Sketch Tools Help
Select Board

harpinblend_single_cam.ino
119
120 void updateLCD(String line1, String line2) {
121     lcd2.clear();
122     lcd2.setCursor(0, 0);
123     lcd2.print(line1);
124     lcd2.setCursor(0, 1);
125     lcd2.print(line2);
126 }
127
128 void controlTraffic(int side) {
129     if (side == 1) {
130         digitalWrite(REDD1, LOW);
131         digitalWrite(GREEN1, HIGH);
132         digitalWrite(YELLOW1, LOW);
133         digitalWrite(REDD2, HIGH);
134         digitalWrite(GREEN2, LOW);
135         digitalWrite(YELLOW2, LOW);
136         updateLCD("Lane 1: GO", "Lane 2: STOP");
137     } else if (side == 2) {
138         digitalWrite(REDD1, HIGH);
139         digitalWrite(GREEN1, LOW);
140         digitalWrite(YELLOW1, LOW);
141         digitalWrite(REDD2, LOW);
142         digitalWrite(GREEN2, HIGH);
143         digitalWrite(YELLOW2, LOW);
144         updateLCD("Lane 1: STOP", "Lane 2: GO");
145     }
146
147     else {
148         digitalWrite(REDD1, LOW);
149         digitalWrite(GREEN1, LOW);
150         digitalWrite(YELLOW1, HIGH);
151         digitalWrite(REDD2, LOW);
152         digitalWrite(GREEN2, LOW);
153     }
154 }
Ln 105, Col 76 X No board selected

```



## **CHAPTER 11**

### **CONCLUSION**

In conclusion, the proposed system presents a comprehensive and innovative solution to address road safety concerns, particularly in mountainous regions. By emulating a ghat environment and utilizing tools such as YOLO for object detection, the system lays the foundation for preventing and detecting accidents in challenging terrains. The potential for optimization and expansion using real-world data and images underscores the adaptability and scalability of the proposed solution. The extension of the project to cover regular city roads and the incorporation of a traffic alert system further demonstrate the versatility and broader applicability of the proposed system. The inclusion of advanced technologies and techniques reflects a commitment to staying at the forefront of road safety initiatives.

The main objective of the system—to automate accident prevention, detection, and emergency response—addresses a critical need, particularly in regions with a high incidence of accidents. The emphasis on timely communication with emergency services aligns with the urgent requirement to reduce fatality rates by ensuring swift assistance. The statistics highlighting the prevalence of accidents, as seen in India, emphasize the pressing need for such systems. The proposed solution not only seeks to prevent accidents but also addresses the significant issue of delayed emergency response, contributing to a safer and more responsive transportation infrastructure. In essence, the proposed system is not only technologically advanced but also socially impactful. It represents a meaningful step towards creating a safer road environment, reducing accidents, and ultimately saving lives. As technology continues to evolve, systems like these are essential for ushering in a new era of road safety and transportation efficiency.

## CHAPTER - 12

### REFERENCES

- [1] D. Mitchell, J. Blanche, O. Zaki, J. Roe, L. Kong, S. Harper, V. Robu, T. Lim, and D. Flynn, “Symbiotic system of systems design for safe and resilient autonomous robotics in offshore wind farms,” *IEEE Access*, vol. 9, pp. 141421–141452, 2021.
- [2] E. Elbasi, N. Mostafa, Z. AlArnaout, A. I. Zreikat, E. Cina, G. Varghese, A. Shdefat, A. E. Topcu, W. Abdelbaki, S. Mathew, and C. Zaki, “Artificial intelligence technology in the agricultural sector: A systematic literature review,” *IEEE Access*, vol. 11, pp. 171–202, 2023.
- [3] S. Yang, J. Ji, H. Cai, and H. Chen, “Modeling and force analysis of a harvesting robot for button mushrooms,” *IEEE Access*, vol. 10, pp. 78519–78526, 2022.
- [4] J. Pak, J. Kim, Y. Park, and H. I. Son, “Field evaluation of path-planning algorithms for autonomous mobile robot in smart farms,” *IEEE Access*, vol. 10, pp. 60253–60266, 2022.
- [5] S. J. LeVoor, P. A. Farley, T. Sun, and C. Xu, “High-accuracy adaptive low-cost location sensing subsystems for autonomous rover in precision agriculture,” *IEEE Open J. Ind. Appl.*, vol. 1, pp. 74–94, 2020.
- [6] N. N. Misra, Y. Dixit, A. Al-Mallahi, M. S. Bhullar, R. Upadhyay, and A. Martynenko, “IoT, big data, and artificial intelligence in agriculture and food industry,” *IEEE Internet Things J.*, vol. 9, no. 9, pp. 6305–6324, May 2022. Smart Farming is Key for the Future of Agriculture | FAO, FAO, Rome, Italy, 2017.
- [7] Mathanraj s1, Akshay k2, Muthukrishnan R3, “Animal Repellent System for Smart Farming Using AI and Deep Learning,” *IRJET* vol 9, 5| May 2022.
- [8] B. Singh, M. Kaur, S. Soni, J. Singh, A. Kumar, and A. Das, “Spatiotemporal mapping of green house gas emission in urban settings using a vehicle mounted IoT enabled pollution sensing modules,” in *Proc. 4th Int. Conf. Inf. Syst. Comput. Netw. (ISCON)*, Nov. 2019, pp. 366–369.
- [9] A. Z. M. T. Kabir, A. M. Mizan, N. Debnath, A. J. Ta-Sin, N. Zinnurayen, and M. T. Haider, “IoT based low cost smart indoor farming management system using an assistant robot and mobile app,” in *Proc. 10th Electr. Power, Electron., Commun., Controls Informat. Seminar (EECCIS)*, Aug. 2020, pp. 155–158.

- [10] N. N. Misra, Y. Dixit, A. Al-Mallahi, M. S. Bhullar, R. Upadhyay, and A. Martynenko, “IoT, big data, and artificial intelligence in agriculture and food industry,” *IEEE Internet Things J.*, vol. 9, no. 9, pp. 6305–6324, May 2022.
- [11] Y. Chen and Y. Li, “Intelligent autonomous pollination for future farming—A micro air vehicle conceptual framework with artificial intelligence and human-in-the-loop,” *IEEE Access*, vol. 7, pp. 119706–119717, 2019.
- [12] X. Han, J. A. Thomasson, T. Wang, and V. Swaminathan, “Autonomous mobile ground control point improves accuracy of agricultural remote sensing through collaboration with UAV,” *Inventions*, vol. 5, no. 1, p. 12, Mar. 2020.
- [13] P. D. Rosero-Montalvo, V. C. Erazo-Chamorro, V. F. López-Batista, M. N. Moreno-García, and D. H. Peluffo-Ordóñez, “Environment monitoring of Rose crops greenhouse based on autonomous vehicles with a WSN and data analysis,” *Sensors*, vol. 20, no. 20, p. 5905, Oct. 2020.
- [14] S. P. Adhikari, G. Kim, and H. Kim, “Deep neural network-based system for autonomous navigation in paddy field,” *IEEE Access*, vol. 8, pp. 71272–71278, 2020.