A Project Report on

SMART HELMET ENFORCEMENT SYSTEM USING YOLOv8 AND IOT

Submitted in partial fulfillment of the requirements for the award of the Degree of

Bachelor of Technology

in

Computer Science and Engineering

By

Pula Vaishnavi	20A91A0552
Namala Aakash	20A91A0544
Karumanchi Hemanth	20A91A0526
Thirupathi Krishna Maheedhar	20A91A0559

Under the Esteemed Supervision of

Dr. S. Rama Sree., Ph.D
Professor & Dean (Academics)

Department of Computer Science and Engineering

ADITYA ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Affiliated to JNTUK Kakinada, Accredited by NBA & NAAC with ‘A++’ Grade)

Aditya Nagar, ADB Road, Surampalem

2020 – 2024

A Project Report on

SMART HELMET ENFORCEMENT SYSTEM USING YOLOv8 AND IOT

Submitted in partial fulfillment of the requirements for the award of the Degree of

Bachelor of Technology

in

Computer Science and Engineering

By

Pula Vaishnavi	20A91A0552
Namala Aakash	20A91A0544
Karumanchi Hemanth	20A91A0526
Thirupathi Krishna Maheedhar	20A91A0559

Under the Esteemed Supervision of

Dr. S. Rama Sree.,Ph.D
Professor & Dean (Academics)

Department of Computer Science and Engineering

ADITYA ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Affiliated to JNTUK Kakinada, Accredited by NBA & NAAC with ‘A++’ Grade)

Aditya Nagar, ADB Road, Surampalem

2020 – 2024

ADITYA ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Affiliated to JNTUK Kakinada, Accredited by NBA & NAAC with ‘A++’ Grade)

Aditya Nagar, ADB Road, Surampalem

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project work entitled “**SMART HELMET ENFORCEMENT SYSTEM USING YOLOv8 AND IOT”** is being submitted by

Pula Vaishnavi	**20A91A0552**
Namala Aakash	**20A91A0544**
Karumanchi Hemanth	**20A91A0526**
Thirupathi Krishna Maheedhar	**20A91A0559**

in partial fulfillment of the requirements for the award of degree of **B.Tech** in Computer Science and Engineering from **Jawaharlal Nehru Technological University Kakinada** is a record of bonafide work carried out by them at **Aditya Engineering College**

The results embodied in this Project report have not been submitted to any other University or Institute for the award of any degree or diploma.

PROJECT GUIDE	**HEAD OF THE DEPARTMENT**
Dr. S. Rama Sree.,Ph.D	**Dr. A. Vanathi, M.E., Ph.D,**
Professor & Dean (Academics)	**Associate Professor**

EXTERNAL EXAMINER

DECLARATION

We hereby declare that the project entitled **"SMART HELMET ENFORCEMENT SYSTEM USING YOLOv8 AND IOT"** is a genuine project. This work has been submitted to the **ADITYA ENGINEERING COLLEGE,** Surampalem, permanently affiliated to **JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, KAKINADA** in partial fulfillment of the **B.Tech** degree**.** We further declare that this project work has not been submitted in full or part of the award for any degree of this or any other educational institutions.

By

Pula Vaishnavi	**(20A91A0552)**
Namala Aakash	**(20A91A0544)**
Karumanchi Hemanth	**(20A91A0526)**
Thirupathi Krishna Maheedhar	**(20A91A0559)**

ACKNOWLEDGEMENT

It is with immense pleasure that we would like to express our indebted gratitude to our project guide **Dr. S. Rama Sree.,Ph.D Professor & Dean (Academics), Department of CSE** who has guided us a lot and encouraged us in every step of the project work, his valuable moral support and guidance throughout the project helped us to a greater extent.

Our sincere thanks to project coordinator **Dr.V.Ravi Kishor M.Tech., Ph.D, Associate Professor, Department of CSE** for providing a great support and suggestions during our project work.

Our deepest thanks to our HOD **Dr. A. Vanathi, M.E., Ph.D, Associate Professor** for inspiring us all the way and for arranging all the facilities and resources needed for our project.

We wish to thank **Dr. S. Rama Sree, Professor** in CSE and Dean (Academics) for her support and suggestions during our project work.

We owe our sincere gratitude to **Dr. M. Sreenivasa Reddy, Principal** for providing a great support and for giving us the opportunity of doing the project.

We are thankful to our **College Management** for providing all the facilities in time to us for completion of our project.

Not to forget, **Faculty, Lab Technicians, Non-teaching staff and our friends** who have directly or indirectly helped and supported us in completing our project in time.

ADITYA ENGINEERING COLLEGE (A)
Aditya Nagar, ADB Road, Surampalem

VISION & MISSION OF THE INSTITUTE

Vision:

To emerge as a premier institute for quality technical education and innovation.

Mission:

M1: Provide learner centric technical education towards academic excellence

M2: Train on technology through collaborations

M3: Promote innovative research & development

M4: Involve industry institute interaction for societal needs

[illegible]

PRINCIPAL

PRINCIPAL
ADITYA ENGINEERING COLLEGE
SURAMPALEM - 533 437

ADITYA ENGINEERING COLLEGE (A)
Aditya Nagar, ADB Road, Surampalem
Department of Computer Science and Engineering

VISION & MISSION OF THE DEPARTMENT

VISION:

To emerge as a competent Centre of excellence in the field of Computer Science and Engineering for industry and societal needs.

MISSION:

- Impart quality and value based education.
- Inculcate the inter personal skills and professional ethics.
- Enable research through state-of-the-art infrastructure.
- Collaborate with industries, government and professional societies.

[illegible]
Head of the Department

Head of the Department
Department of CSE
ADITYA ENGINEERING COLLEGE (A

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

Graduates of the Program will

- **PEO 1:** Adopt new technologies and provide innovative solutions.
- **PEO 2:** Be employable, become an entrepreneur or researcher for a successful career.
- **PEO 3:** Demonstrate interpersonal, multi-disciplinary skills and professional ethics to serve society.

[illegible]

Head of the Department

Head of the Departmen
Department of CSE
DITYA ENGINEERING COLLEGE (A

PROGRAM OUTCOMES (POs)

After successful completion of the program, the graduates will be able to

PO 1 **Engineering Knowledge:** Apply knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

PO 2 **Problem Analysis:** Identify, formulate, research literature and analyze complex engineering problems, reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.

PO 3 **Design/Development of Solutions:** Design solutions for complex engineering problems and design systems, components or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal, and environmental considerations.

PO 4 **Conduct Investigations of Complex Problems:** Conduct investigations of complex problems using research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of information to provide validconclusions.

PO 5 **Modern Tool Usage:** Create, select and apply appropriate techniques, resources, and modern engineering and IT tools, including prediction and modelling, to complex engineering activities, with an understanding of the limitations.

PO 6 **The Engineer and Society:** Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to professional engineering practice.

PO 7 **Environment and Sustainability:** Understand the impact of professional engineering solutions in societal and environmental contexts and demonstrate knowledge of, and need for sustainable development.

PO 8 **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of engineering practice.

PO 9 **Individual and Teamwork:** Function effectively as an individual, and as a member or leader in diverse teams and in multidisciplinary settings.

PO 10 **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO 11 **Project Management and Finance:** Demonstrate knowledge and understanding of engineering management principles and apply these to one's own work, as a member and leader in a team and to manage projects in multidisciplinary environments.

PO 12 **Life-Long Learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

[illegible]

Head of the Department

Head of the Department
Department of CSE
DITYA ENGINEERING COLLEGE (A

ADITYA ENGINEERING COLLEGE (A)
Aditya Nagar, ADB Road, Surampalem
Department of Computer Science and Engineering

PROGRAM SPECIFIC OUTCOMES (PSOs)

After successful completion of the program, the graduates will be able to

PSO 1: Develop efficient solutions to real world problems using the domains of Algorithms, Networks, database management and latest programming tools and techniques.

PSO 2: Provide data centric business solutions through emerging areas like IoT, AI , data analytics and Block Chain technologies.

[illegible]
Head of the Department
Head of the Departmen
Department of CSE
DITYA ENGINEERING COLLEGE (A

ABSTRACT

In response to the escalating concern of road accidents, particularly among motorcyclists, "Helmet Detection for Bike Automation" emerges as a potent solution. The project aims to mitigate the risks associated with improper helmet usage, a leading cause of fatal injuries in motorcycle accidents. Leveraging the power of Deep Learning techniques, specifically YOLO, the system provides real-time detection of helmets. Integrated with a Raspberry Pi unit, OpenCV, LEDs, a buzzer, and an LCD display, the system offers a comprehensive approach to enhancing road safety.

The workflow of the system is straightforward yet effective. Upon ignition, the system prompts the motorcyclist to confirm helmet presence with a simple button press. In the absence of a detected helmet, the system triggers visual and auditory warnings, urging immediate corrective action. A critical 30-second window is provided for the rider to comply. If no helmet is detected within this timeframe, the system intervenes decisively by automatically shutting down the motorcycle.

The benefits of this innovative system extend far beyond its technological prowess. "Helmet Detection for Bike Automation" stands as a beacon of hope for motorcyclists, promising a safer and more responsible riding experience. By empowering riders with real-time feedback and ensuring adherence to safety protocols, the project holds the potential to significantly reduce motorcycle-related accidents and save precious lives on the road.

TABLE OF CONTENTS

S.No	INDEX	Page.No
1.	**INTRODUCTION**	**01**
	1.1 Introduction to a project	01
	1.2 Existing System and Disadvantages	02
	1.3 Proposed System and Advantages	02
	1.4 Objectives of the Project	05
	1.5 Organization of the project	05
2.	**REQUIREMENTS ANALYSIS**	**07**
	2.1 Introduction of Requirement Analysis	07
	2.2 Hardware& Software Requirements	07
	2.3 Datasets	29
	2.4 Software Requirements Specification	30
	2.5 Societal need	30
3.	**LITERATURE SURVEY**	**31**
	3.1 Related Works	31
	3.2 Survey Table	34
4.	**MODULES**	**37**
	4.1 Introduction to Modules	37
	4.2 Module Implementation	37
5.	**SYSTEM DESIGN**	**41**
	5.1 Introduction to System Design	41
	5.2 Project Flow of the Project	42
	5.3 Flowchart Diagram	42
	5.4 UML Diagrams	43

6. SYSTEM IMPLEMENTATION 49
6.1 Introduction To System Implementation 49
6.2 Implementation of Selected Software & Programming Language 49
6.3 Implementation of Algorithm 50
6.4 Final Code for Raspberry Board 51
6.5 Final implementation 54
7. TESTING 55
7.1 Introduction 55
7.2 Test cases 56
8. SCREENS & RESULTS 59
9. CONCLUSION and FUTURE SCOPE 66
9.1 Conclusion 66
9.2 Future Scope 66
10. BIBLIOGRAPHY 67
10.1 References 67
10.2 Web Links 68

LIST OF FIGURES

Figure.No	Name of the Figure	Page.No
1.1	Bounding box	4
1.2	Working YOLOv8	4
1.3	Yolo Object Detection Models Timeline	4
2.1	Raspberry Pi 4	8
2.2	BCM2711B0	9
2.3	Dual micro-HDMI	9
2.4	Gigabit Ethernet and USB 3.0 ports	9
2.5	Pi Camera port	10
2.6	Pi Zero	11
2.7	LCD 16×2	12
2.8	LCD 16 X 2 Pins	13
2.9	Piezo Buzzer	15
2.10	Different Views of a Piezo Buzzer	15
2.11	PCB of a Piezo Buzzer	15
2.12	Electronic Components behind PCB of Piezo Buzzer	16
2.13	Diaphragm Attached to PCB the Arrangement Produces Sound	16
2.14	Metal Plate Behind the Diaphragm	16
2.15	Piezo ceramic Disc and Attached Metal Plate	17
2.16	Working Principle of Piezo Buzzer	17
2.17	LED’s	19
2.18	Switch	19
2.19	SPST working	20
2.20	DC motor	21
2.21	Block Diagram of Power Supply	22
2.22	Bridge flow	23
2.23	Output Waveform of DC	23
2.24	Regulator	24
2.25	Circuit Diagram of Power Supply	25
2.26	Model Code	26

2.27	Pandas	27
2.28	Ultralytics	27
2.29	Flask	28
2.30	Dataset	29
5.1	System Architecture	41
5.2	Flow Chart for the Project Flow	42
5.3	Use case diagram of our proposed project	43
5.4	Class diagram of our proposed project	44
5.5	Sequence diagram of our proposed project	45
5.6	Collaboration diagram of our proposed project	45
5.7	State Chart Diagram of our proposed project	46
5.8	Activity Diagram of our proposed project	47
5.9	Component Diagram of our proposed project	47
5.10	ER Diagram of our proposed project	48
6.1	Flow of Implementation	51

LIST OF TABLES

Table.No	Name of the Table	Page.No
2.1	Pin Configuration of the Buzzer	18
3.1	Survey Table	34
7.1	Test Case – 1	56
7.2	Test Case – 2	57
7.3	Test Case – 3	58

LIST OF SCREENS

Table.No	Name of the Screen	Page.No
8.1	Dataset 1	59
8.2	Dataset 2	59
8.3	Model Code	60
8.4	Installing Ultralytics	60
8.5	Training the model	61
8.6	Training with collected datasets	61
8.7	Running Epoch	62
8.8	Training visualization	62
8.9	Prototype	63
8.10	Detection with helmet	63
8.11	wheel starts rotating after detecting the helmet	64
8.12	Detection without helmet	64
8.13	Time indication for 10 seconds	65
8.14	Wheel stops after the completion of time indication	65

1. INTRODUCTION

Deep learning, a subset of machine learning and artificial intelligence, emulates the human process of acquiring specific knowledge. Within this realm, Convolutional Neural Networks (CNNs) structured You Only Look Once (YOLO) play a pivotal role, particularly in image and video analysis. In our proposed model, we leverage YOLO to recognize whether a driver is wearing a helmet or not. Rigorous training of image datasets, encompassing both helmeted and non-helmeted scenarios, equips the model with the ability to process real-time camera feeds from Air Board, producing accurate results on the presence or absence of a helmet. The model's algorithm enables the classification of diverse image categories, contributing to a robust framework for enhancing road safety.

1.1 Introduction to project

The proposed model represents a significant advancement in the realm of road safety by harnessing the power of deep learning, particularly through Convolutional Neural Networks (CNNs) with the You Only Look Once (YOLO) architecture. By employing YOLO, the model is capable of real-time object detection, specifically focusing on the crucial task of recognizing whether a driver is wearing a helmet. This application of deep learning is pivotal in mitigating risks associated with two-wheeler transportation, where helmet usage is a critical safety measure. Through rigorous training on diverse image datasets encompassing both helmeted and non-helmeted scenarios, the model becomes adept at accurately processing real-time camera feeds, enabling it to provide timely warnings to riders who neglect to wear helmets.

Python programming serves as the backbone for implementing this advanced model, providing a flexible and powerful framework for developing and deploying deep learning solutions. Leveraging the extensive ecosystem of Python libraries and tools, such as TensorFlow, PyTorch, and OpenCV, facilitates efficient model development and integration with real-world applications. Moreover, the incorporation of a safety intervention mechanism demonstrates a proactive approach to enforcing helmet usage. By utilizing a Raspberry Pi connected to hardware components like relay switches and DC motors, the model can actively intervene in instances of persistent disregard for helmet safety, thereby reinforcing the importance of adhering to safety protocols while riding motorcycles.

Beyond its immediate application in helmet detection, the proposed model contributes to fostering a safety-conscious motorcycle riding culture and creating a road environment that is secure and accident-free. By integrating cutting-edge technologies like machine learning, image processing, and computer vision, the model addresses critical safety concerns and promotes responsible behavior among riders. This innovative approach not only enhances individual safety but also aligns with broader societal objectives of reducing road accidents and minimizing the associated human and economic costs. Ultimately, the convergence of deep learning algorithms, Python programming, and hardware intervention mechanisms represents a paradigm shift in promoting road safety and underscores the transformative potential of technology in safeguarding human lives.

1.2 Existing System

The prevailing scenario witnesses a distressing trend of inadequate helmet usage among motorcyclists, leading to a surge in road accidents and fatalities. The absence of a reliable mechanism to enforce helmet safety poses a considerable risk to riders' well-being.

Many systems are available to detect whether the rider is wearing a helmet or not. However, in these systems, detection and arrangements are integrated into the rider's helmet using IR sensors and other sensors. The rider is required to use the specific helmet equipped with these sensors exclusively while riding the bike.

Disadvantages:

- The rider must wear only the helmet with sensors.
- There may be potential harm due to sensors present at the head.
- The rider cannot wear other helmets.

1.3 Proposed System

"Helmet Detection for Bike Automation" presents a robust and intelligent solution to the prevalent issue of improper helmet usage. Through the implementation of Deep Learning techniques, specifically YOLO, the system achieves unparalleled accuracy in detecting helmet presence. The Raspberry Pi unit serves as the central processing hub, orchestrating the interactions between the camera module, OpenCV, LEDs, buzzer, and LCD display.

The operational workflow of the system is designed for seamless user experience. Upon ignition, the motorcyclist is prompted to confirm helmet presence with a single button press. In the event of a missing helmet, the system promptly alerts the rider through a combination of visual LED indicators and an audible buzzer. A grace period of 30 seconds is provided for the rider to rectify the situation. Should no helmet be detected within this timeframe, the system takes proactive measures by automatically shutting down the motorcycle.

Advantages:

- **Enhanced Road Safety:** Ensures strict adherence to helmet safety protocols, minimizing the risk of head injuries.
- **Real-time Detection:** Utilizes Deep Learning techniques for instantaneous and accurate helmet detection.
- **Immediate Alerts:** Visual LED indicators and an audible buzzer prompt immediate action from the rider.
- **Automatic Shutdown:** Prevents motorcycle operation in the absence of a detected helmet, prioritizing rider safety.

You Only Look Once (YOLO):

In the proposed model aimed at enhancing road safety for two-wheeler transportation, the You Only Look Once (YOLO) architecture operates as follows: Upon receiving input from real-time camera feeds, YOLO divides the input image into a grid and predicts bounding boxes and class probabilities for objects within each grid cell simultaneously. This single forward pass through the neural network enables YOLO to achieve real-time object detection with remarkable speed and accuracy. In the context of helmet detection, YOLO is trained on a dataset containing annotated images of both helmeted and non-helmeted scenarios, allowing the model to learn distinctive features associated with helmets. During inference, YOLO efficiently processes incoming video frames, swiftly identifying the presence or absence of helmets worn by drivers. This streamlined approach to object detection, coupled with YOLO's capability to handle complex scenes and varying lighting conditions, enables the model to deliver reliable results in real-world settings. By leveraging YOLO's efficient and accurate object detection capabilities, the model effectively addresses critical safety concerns by providing timely warnings to riders who neglect helmet usage and facilitating proactive interventions to reinforce the importance of wearing helmets for a secure ride.

In the context of autonomous vehicles, YOLO (You Only Look Once) can be used for object detection, which is a crucial task for the vehicle to understand its environment and make informed decisions. Here's how YOLO could work in an object detection system for autonomous vehicles:

Input Data:
YOLO takes an image or a video frame as input.

Bounding Box Prediction:
YOLO divides the input image into a grid and predicts bounding boxes and class probabilities for each grid cell. For each bounding box, YOLO predicts the coordinates (x, y) of the box's center, the width, and the height. Additionally, it predicts the probability of the presence of an object and the class of the object.

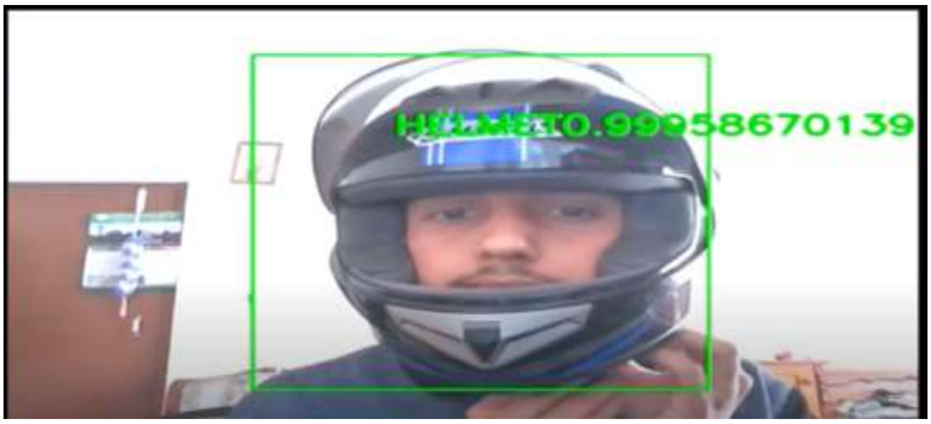


Fig 1.1 Bounding box

Multiple Objects:
YOLO is capable of detecting multiple objects in a single pass. Each bounding box can potentially represent a different object.

Non-Maximum Suppression (NMS):
After the predictions, a post-processing step is performed to remove duplicate and low-confidence detections using NMS. This ensures that each object is represented by a single bounding box with high confidence.

Output:
The final output consists of bounding boxes with their associated class labels and confidence scores.

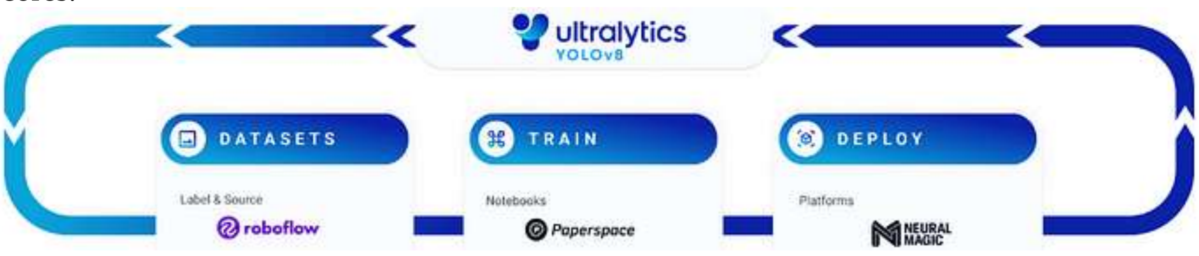


Fig 1.2 Working of YOLOv8

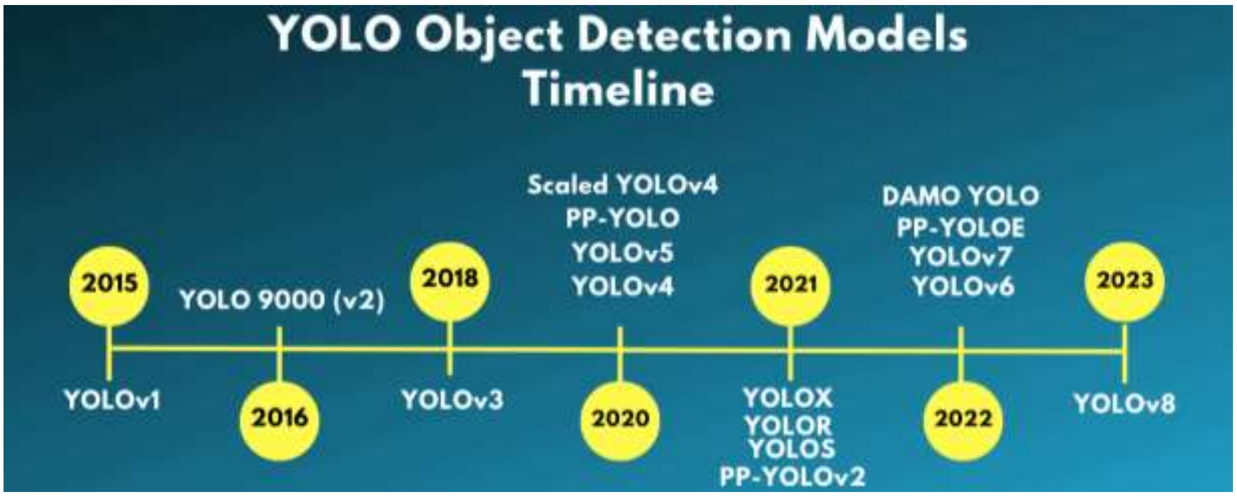


Fig 1.3 Yolo Object Detection Models Timeline

1.4 Objectives of the Project

The objective of the context provided is to present a proposed model that utilizes advanced technologies, such as deep learning, computer vision, and hardware intervention, to enhance road safety for two-wheeler transportation. Specifically, the model aims to address the critical safety concern of helmet usage among motorcycle riders. By leveraging the You Only Look Once (YOLO) architecture, the model seeks to detect in real-time whether a driver is wearing a helmet or not by processing live camera feeds. The overall goal is to promote a safety-conscious motorcycle riding culture and reduce the risk of accidents by encouraging and enforcing the use of helmets through timely warnings and interventions. Additionally, the model aims to demonstrate the transformative potential of technology in mitigating road safety hazards and fostering a secure road environment.

1.5 Organization of the Project

Introduction

This section introduces the model for detection of helmet and automation in two-wheelers using the Deep Learning technique You Only Look Once (YOLO) and IOT.

Requirement Analysis

This section focuses on the hardware and software requirements that support the accomplishment of the project successfully.

Literature Survey

This section aims at different proposes that are related in various kinds of detection models includes the helmet detection.

Modules

This section focuses on various modules the project is divided, which involves dataset collection, develop model, components inter-connection, detect etc.

System Design

This section describes the designs related to the processes involved in detecting the presence of helmet and automating the two-wheeler using UML diagrams.

System Implementation

This section focuses on implementing the modules that are involved in the process using python and also the IoT to implement the project successfully.

Testing

In this section after implementing the project various tests and need of testing is described. The test cases related to different inputs are described briefly.

Screens & Reports

This section contains set of results and reports of the project. Screenshots of the process and outputs are given**.**

Conclusion & Future Scope

This section highlights the conclusions from the project and describes about future enhancements that have scope to be implemented.

Bibliography

This section contains references from different sources.

2. REQUIREMENT ANALYSIS

2.1 Introduction to Requirement Analysis

Requirement Analysis is a systematic process that lays the groundwork for successful software development by identifying and documenting the needs and expectations for implementing the project and defining the scope of the system, and providing guidance. It is a critical phase that significantly influences the success of the process and also the implementation can be done in a good way. These are the necessary tools that are required to initiate a project. These are both hardware as well as software tools.

2.2 Hardware and Software Requirements

Hardware Requirements

- Raspberry Pi 4
- Camera Module
- LEDs
- Buzzer
- Button
- LCD Display (16 X 2)
- DC Motor

Software Requirements

- Deep Learning (YOLO)
- Python (OpenCV)
- Raspberry Pi OS
- IDE: Visual Studio Code, Google Co-lab
- Libraries: NumPy, Pandas, Ultralytics
- Framework: Flask

Raspberry Pi :-

Raspberry Pi 4 has a full-chip redesign, the first in the history of Raspberry Pi, and it's unlocked new levels of performance. Our specs & benchmarks show just how much faster Raspberry Pi 4.

Build the home of the future in the latest edition of The MagPi magazine. We go around the house, room-by-room, and add super-smart home improvements with Raspberry Pi.

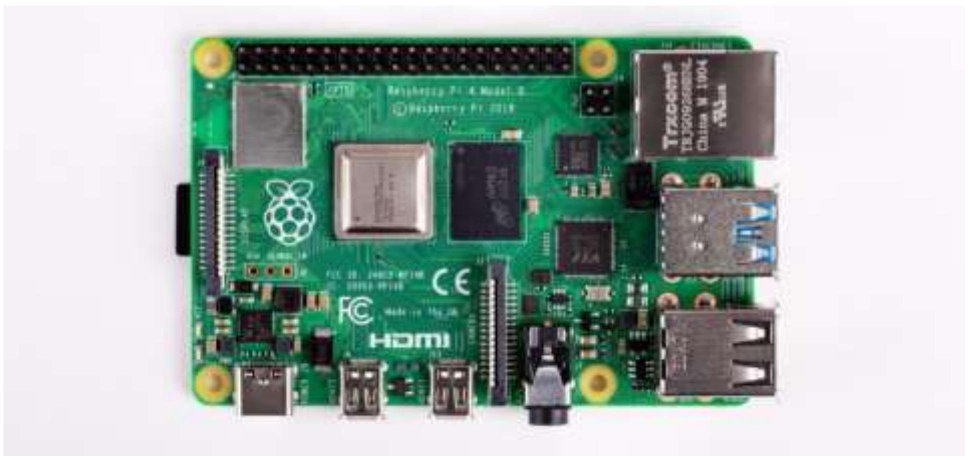


Fig 2.1 Raspberry Pi 4

Raspberry Pi 4 improves on its predecessor, with improved specifications across the board. Our Raspberry Pi 4 benchmark tests show a huge increase in performance over previous models.

It's not hard to see where this benchmark boost comes from. The brand-new BCM2711B0 system-on-chip has more powerful processing cores, the first upgrade to the graphics processor in the history of the project, and vastly improved bandwidth for both memory and external hardware.

Gone is the single-lane USB bottleneck which hampered performance on older models, and Raspberry Pi 4 shines in benchmarks as a result.

Raspberry Pi 4 specs

- **SoC:** Broadcom BCM2711B0 quad-core A72 (ARMv8-A) 64-bit @ 1.5GHz
- **GPU:** Broadcom Video Core VI
- **Networking:** 2.4 GHz and 5 GHz 802.11b/g/n/ac wireless LAN
- **RAM:** 1GB, 2GB, or 4GB LPDDR4 SDRAM
- **Bluetooth:** Bluetooth 5.0, Bluetooth Low Energy (BLE)
- **GPIO:** 40-pin GPIO header, populated
- **Storage:** microSD
- **Ports:** 2 × micro-HDMI 2.0, 3.5 mm analogue audio-video jack, 2 × USB 2.0, 2 × USB 3.0, Gigabit Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)
- **Dimensions:** 88 mm × 58 mm × 19.5 mm, 46 gBrand new SoC: BCM2711B0, quad-core 1.5GHz

The new BCM2711B0 system-on-chip offers an impressive performance boost over its predecessors.

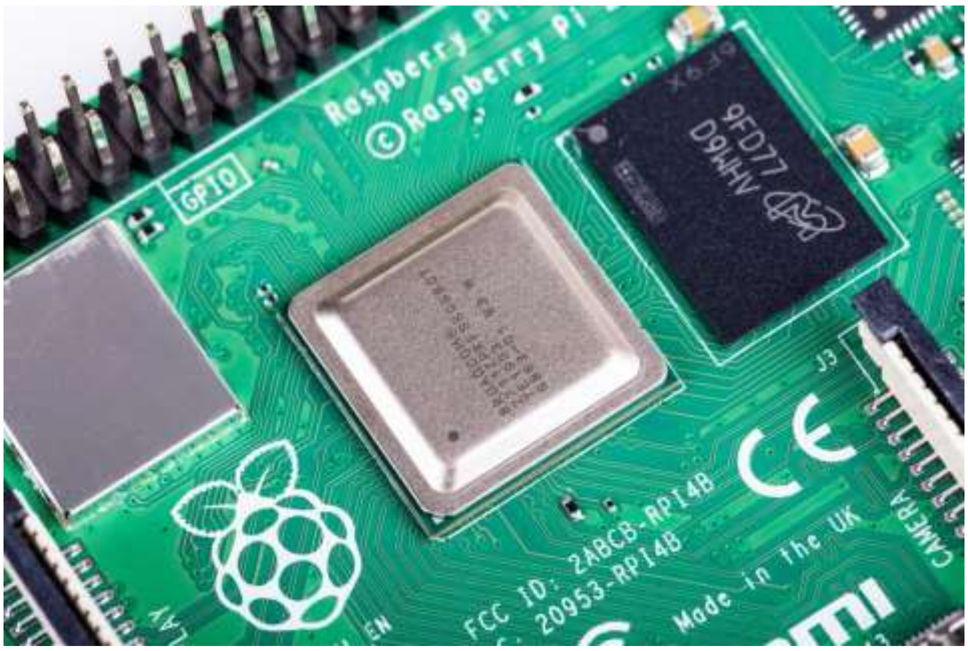


Fig 2.2 BCM2711B0

Dual display via micro-HDMI
The two micro-HDMI connectors enable Raspberry Pi 4 to drive two 4K displays at up to 4Kp30, or a single display at up to 4Kp60

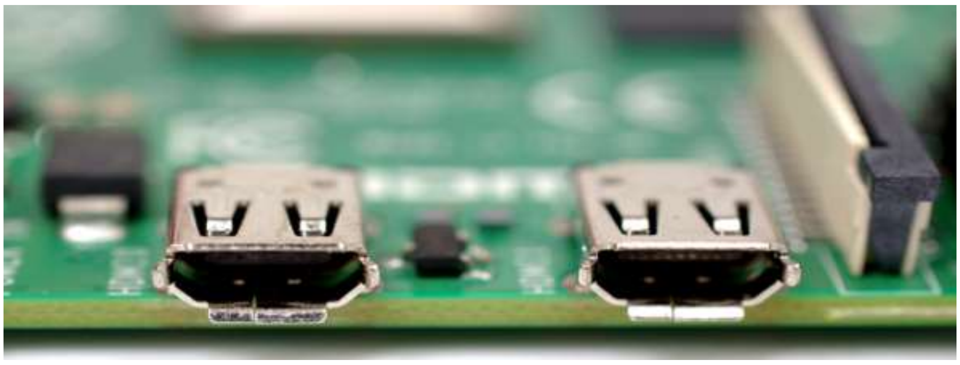


Fig 2.3 Dual micro-HDMI

Gigabit Ethernet and USB 3.0
The Ethernet port, relocated to the top-right of the board, now offers full-speed network connectivity with no bottlenecks. Two USB 3.0 ports, center, offer high-speed connectivity for external devices including storage and accelerator hardware.

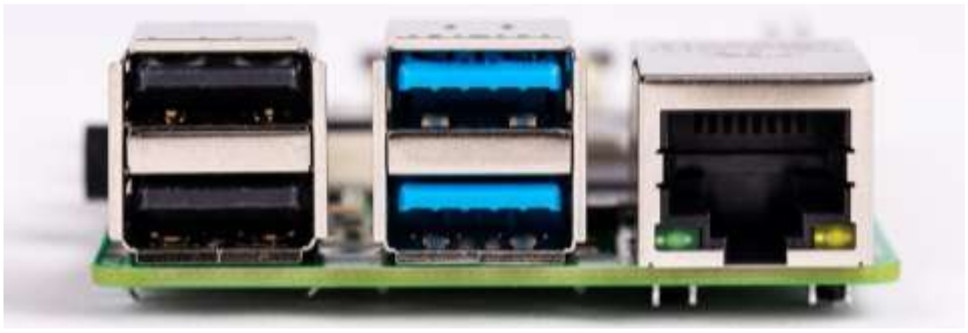


Fig 2.4 Gigabit Ethernet and USB 3.0 ports

PI CAMERA

Make sure your Pi is off while installing the camera module. Although it is possible to install the camera while the Pi is on, this isn't good practice (if the camera is active when removed, it's possible to damage it).

Connect your camera module to the CSI port on your Raspberry Pi; this is the long thin port adjacent to the HDMI socket. Gently lift the collar on top of the CSI port (if it comes off, don't worry, you can push it back in but try to be more gentle in future!). Slide the ribbon cable of the camera module into the port with the blue side facing the Ethernet port (or where the Ethernet port would be if you've got a model A/A+).

Once the cable is seated in the port, press the collar back down to lock the cable in place. If done properly you should be able to easily lift the Pi by the camera's cable without it falling out. The following illustrations show a well-seated camera cable with the correct orientation:

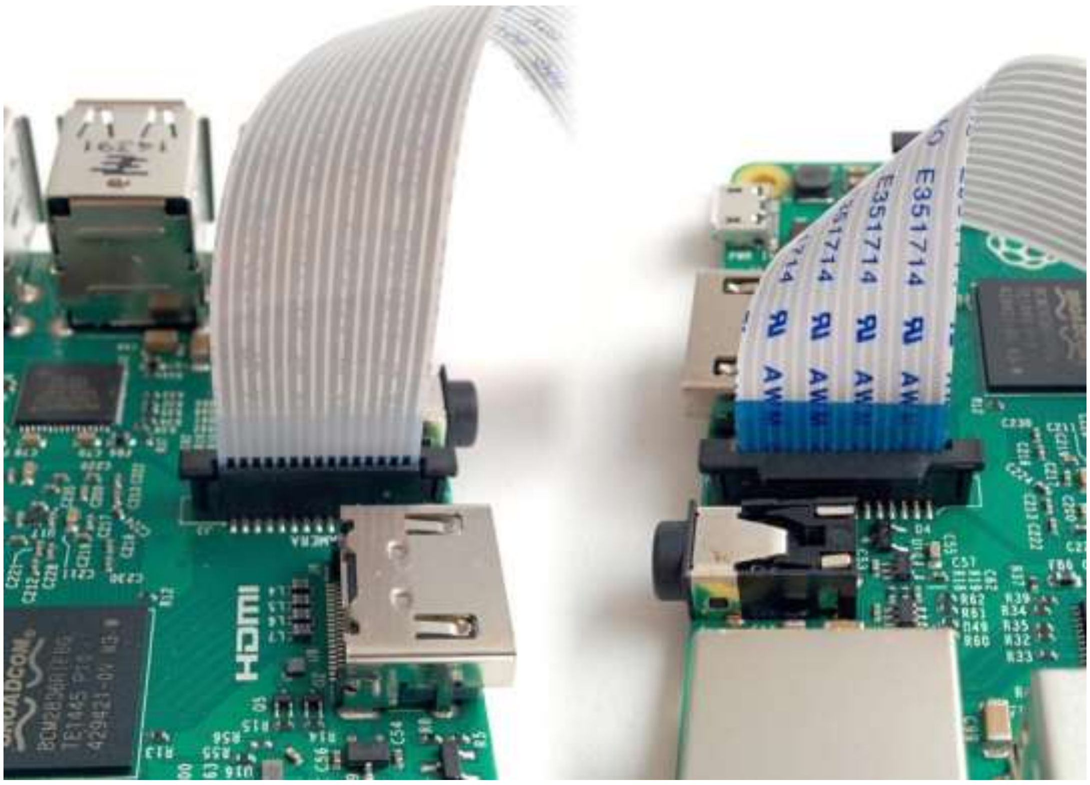


Fig 2.5 Pi Camera port

Make sure the camera module isn't sat on anything conductive (e.g. the Pi's USB ports or its GPIO pins).

Pi Zero

The 1.2 model of the Raspberry Pi Zero includes a small form-factor CSI port which requires a camera adapter cable.

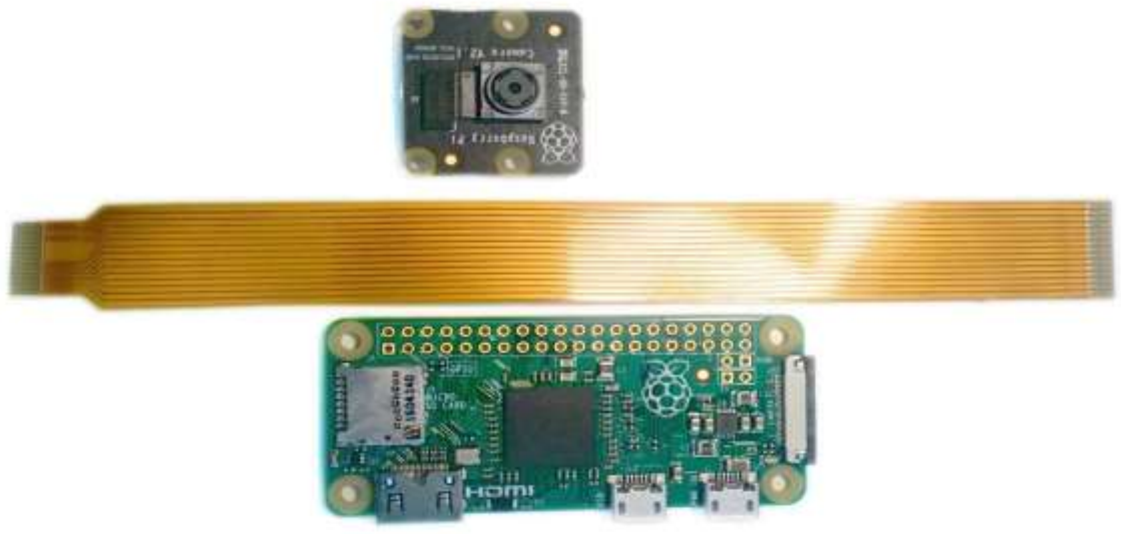


Fig 2.6 Pi Zero

To attach a camera module to a Pi Zero:

1. Remove the existing camera module's cable by gently lifting the collar on the camera module and pulling the cable out.
2. Next, insert the wider end of the adapter cable with the conductors facing in the same direction as the camera's lens.
3. Finally, attach the adapter to the Pi Zero by gently lifting the collar at the edge of the board (be careful with this as they are more delicate than the collars on the regular CSI ports) and inserting the smaller end of the adapter with the conductors facing the back of the Pi Zero.

LCD Display 16 X 2

Nowadays, we always use the devices which are made up of LCDs such as CD players, DVD players, digital watches, computers, etc. These are commonly used in the screen industries to replace the utilization of CRTs. Cathode Ray Tubes use huge power when compared with LCDs, and CRTs heavier as well as bigger. These devices are thinner as well power consumption is extremely less. The LCD 16×2 working principle is, it blocks the light rather than dissipate. This article discusses an overview of LCD 16X2, pin configuration and its working.

What is the LCD 16×2?

The term LCD stands for liquid crystal display. It is one kind of electronic display module used in an extensive range of applications like various circuits & devices like mobile phones, calculators, computers, TV sets, etc. These displays are mainly preferred for multi-segment light-emitting diodes and seven segments. The main benefits of using this module are inexpensive; simply programmable, animations, and there are no limitations for displaying custom characters, special and even animations, etc.

Fig 2.7 LCD 16×2

The 16×2 LCD pinout is shown below.

- Pin1 (Ground/Source Pin): This is a GND pin of display, used to connect the GND terminal of the microcontroller unit or power source.
- Pin2 (VCC/Source Pin): This is the voltage supply pin of the display, used to connect the supply pin of the power source.
- Pin3 (V0/VEE/Control Pin): This pin regulates the difference of the display, used to connect a changeable POT that can supply 0 to 5V.
- Pin4 (Register Select/Control Pin): This pin toggles among command or data register, used to connect a microcontroller unit pin and obtains either 0 or 1(0 = data mode, and 1 = command mode).
- Pin5 (Read/Write/Control Pin): This pin toggles the display among the read or writes operation, and it is connected to a microcontroller unit pin to get either 0 or 1 (0 = Write Operation, and 1 = Read Operation).
- Pin 6 (Enable/Control Pin): This pin should be held high to execute Read/Write process, and it is connected to the microcontroller unit & constantly held high.
- Pins 7-14 (Data Pins): These pins are used to send data to the display. These pins are connected in two-wire modes like 4-wire mode and 8-wire mode. In 4-wire mode, only four pins are connected to the microcontroller unit like 0 to 3, whereas in 8-wire mode, 8-pins are connected to microcontroller unit like 0 to 7.
- Pin15 (+ve pin of the LED): This pin is connected to +5V
- Pin 16 (-ve pin of the LED): This pin is connected to GND.

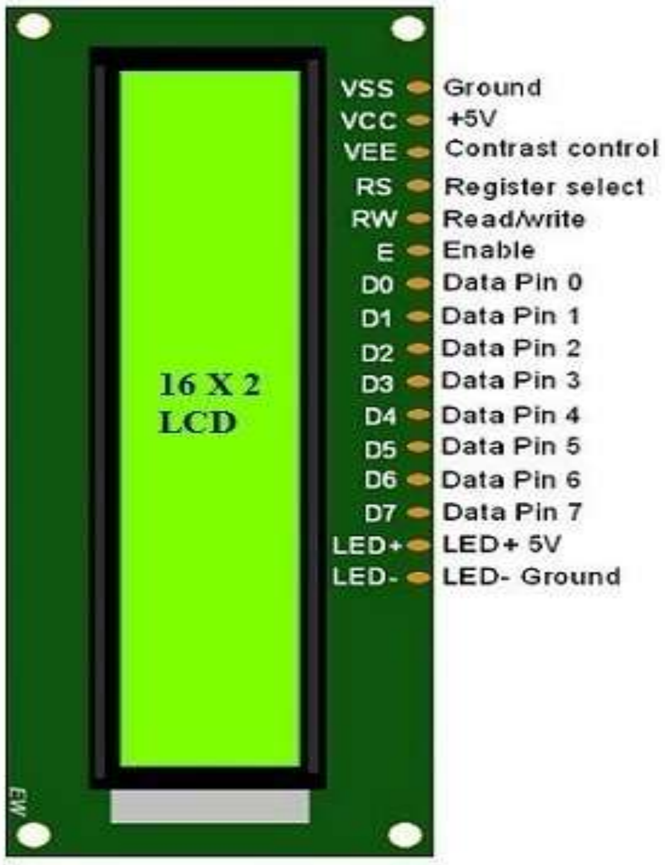


Fig 2.8 LCD 16 X 2 Pins

Features of LCD16x2

The features of this LCD mainly include the following.

- The operating voltage of this LCD is 4.7V-5.3V
- It includes two rows where each row can produce 16-characters.
- The utilization of current is 1mA with no backlight
- Every character can be built with a 5×8-pixel box
- The alphanumeric LCDs alphabets & numbers
- Is display can work on two modes like 4-bit & 8-bit
- These are obtainable in Blue & Green Backlight
- It displays a few custom generated characters

Registers of LCD

A 16×2 LCD has two registers like data register and command register. The RS (register select) is mainly used to change from one register to another. When the register set is ‘0’, then it is known as command register. Similarly, when the register set is ‘1’, then it is known as data register.

Command Register

The main function of the command register is to store the instructions of command which are given to the display. So that predefined tasks can be performed such as clearing the display, initializing, set the cursor place, and display control. Here commands processing can occur within the register.

Data Register

The main function of the data register is to store the information which is to be exhibited on the LCD screen. Here, the ASCII value of the character is the information which is to be exhibited on the screen of LCD. Whenever we send the information to LCD, it transmits to the data register, and then the process will be starting there. When register set =1, then the data register will be selected.

PIEZO BUZZER

A buzzer or beeper is an audio signaling device, which may be mechanical, electromechanical, or piezoelectric (piezo for short). Typical uses of buzzers and beepers include alarm devices, timers, and confirmation of user input such as a mouse click or keystroke. Piezo buzzer is an electronic device commonly used to produce sound. Light weight, simple construction and low price make it usable in various applications like car/truck reversing indicator, computers, call bells etc. It is based on the inverse principle of piezo electricity. It is the phenomena of generating electricity when mechanical pressure is applied to certain materials and the vice versa is also true. Such materials are called piezo electric materials. Piezo electric materials are either naturally available or manmade. Piezo ceramic is class of manmade material, which poses piezo electric effect and is widely used to make disc, the heart of piezo buzzer. When subjected to an alternating electric field they stretch or compress, in accordance with the frequency of the signal thereby producing sound. They generate a loud and sharp sound. So, they are typically used in alarm circuits. Also, they are used to make an alert of an event, signal or sensor input. A special characteristic of piezo buzzer is that, the sound pitch or level is not dependent on the voltage level. i.e., it works only in a specific voltage range. Typically, a piezo buzzer can generate a sound in the range of 2 to 4 kHz. The figure 2.9 below shows a very commonly used piezo buzzer also called piezo transducer operating at DC voltage. Encapsulated in a cylindrical plastic coating. It has a hole on the top face for sound to propagate. A yellow metallic disc which plays an important role in the producing sound can be seen through the hole.

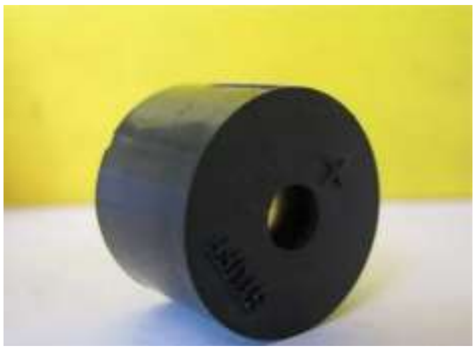


Fig 2.9 A Piezo Buzzer

The figure 2.10 below shows the buzzer at different angles. The two leads are used to supply a DC voltage.

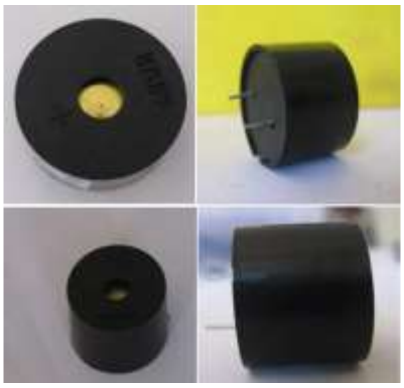


Fig 2.10 Different Views of a Piezo Buzzer

Electronic Components

On removing the back plastic cover the PCB with electronic components soldered on it is shown in the figure 2.11 below.

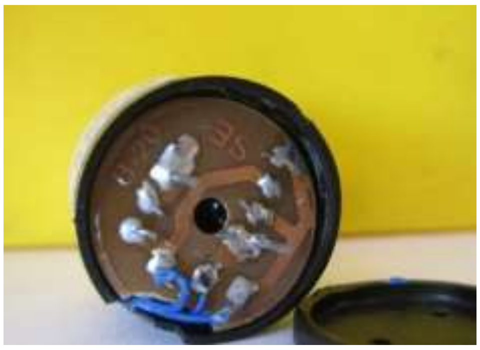


Fig. 2.11 PCB of a Piezo Buzzer

This is the opposite side of the PCB in figure 2.12 below, having the necessary electronic components: a resistor, a transistor and an inductor. The input to the transducer is a low voltage DC signal, however in order to produce sound the piezo ceramic disc needs oscillations of high voltage. The transistor and resistor combination works as an oscillator circuit to produce low amplitude oscillations from the DC voltage. The magnitude of these oscillations is amplified by the inductor.

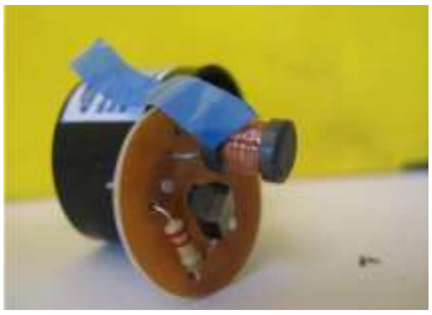


Fig. 2.12 Electronic Components behind PCB of Piezo Buzzer

Diaphragm

The figure 2.13 below shows the circular shaped diaphragm connected to the rest of the electronic components which is the source of producing sound. It consists of a metal plate made up of brass or stainless steel and a piezoceramic disc (white colored disc) of smaller radius connected to each other with a conductive adhesive. The metal plate is used because the resonance frequency of the piezo ceramic material is too high to produce audible sound.

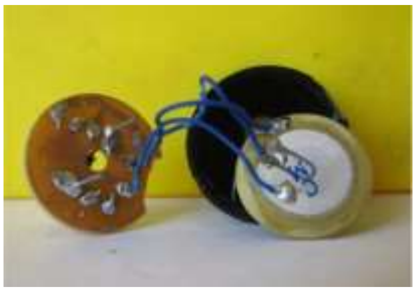


Fig. 2.13 Diaphragm Attached to PCB the Arrangement that Produces Sound

The figure 2.14 below shows what is present behind the diaphragm, which is typically a metal plate.

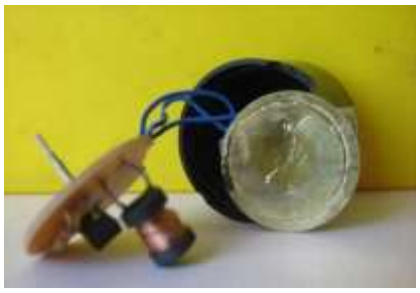


Fig. 2.14 Metal Plate Behind the Diaphragm

Piezoceramic Disc

The figure 2.15 below is a clear view of the piezo ceramic disc glued to the metal plate (a small segment of the piezo ceramic disc has been removed for proper understanding). The piezo ceramic disc is coated by electrodes on both sides. There are three wires connected to the diaphragm. One to the metal plate and one each to the electrodes of the ceramic material. The wire connected to the metal plate is grounded. One electrode is used to provide the high amplitude oscillation signals to the ceramic disc and the second is used to provide feedback to the oscillating circuit.

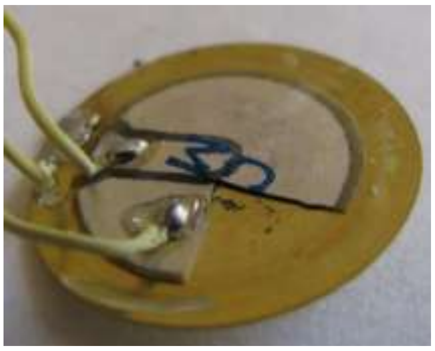


Fig. 2.15 Piezo ceramic Disc and Attached Metal Plate

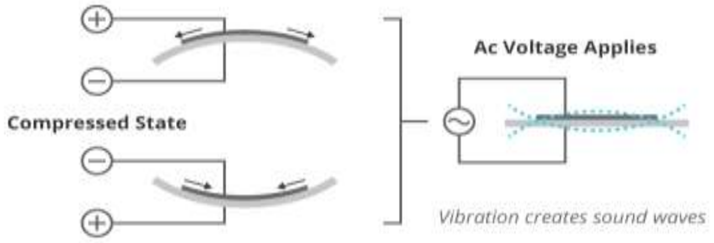


Fig. 2.16 Working Principle of Piezo Buzzer

"Piezoelectricity" is an effect where certain crystals will change shape when you apply electricity to them. By applying an electric signal at the right frequency, the crystal can make sound. When a small DC voltage is applied to the input pins, it is first converted to an oscillating signal using the combination of resistor and transistor. These oscillating signals are amplified using the inductor coil. When high voltage alternating signals are applied to the piezo ceramic disc, it causes mechanical expansion and contraction in radial direction. This causes the metal plate to bend in opposite direction. When metal plate bends and shrinks in opposite direction continuously it produces sound waves in the air.

Table 2.1 Pin Configuration of the Buzzer

Pin Number	**Pin Name**	**Description**
1	Positive	Identified by (+) symbol or longer terminal lead. Can be powered by 6V DC
2	Negative	Identified by short terminal lead. Typically connected to the ground of the circuit

Features and Specifications of the Buzzer

- Rated Voltage: 6V DC
- Operating Voltage: 4-8V DC
- Rated current: <30mA
- Sound Type: Continuous Beep
- Resonant Frequency: ~2300 Hz
- Small and neatly sealed package

LED:

A light-emitting diode (LED) is a semiconductor light source that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons. The color of the light (corresponding to the energy of the photons) is determined by the energy required for electrons to cross the band gap of the semiconductor. White light is obtained by using multiple semiconductors or a layer of light-emitting phosphor on the semiconductor device.

Appearing as practical electronic components in 1962, the earliest LEDs emitted low-intensity infrared light. Infrared LEDs are used in remote-control circuits, such as those used with a wide variety of consumer electronics. The first visible-light LEDs were of low intensity and limited to red. Modern LEDs are available across the visible, ultraviolet, and infrared wavelengths, with high light output.

Early LEDs were often used as indicator lamps, replacing small incandescent bulbs, and in seven-segment displays. Recent developments have produced high-output white light LEDs suitable for room and outdoor area lighting. LEDs have led to new displays and sensors, while their high switching rates are useful in advanced communications technology.

LEDs have many advantages over incandescent light sources, including lower energy consumption, longer lifetime, improved physical robustness, smaller size, and faster switching. LEDs are used in applications as diverse as aviation lighting, automotive headlamps, advertising, general lighting, traffic signals, camera flashes, lighted wallpaper, plant growing light and medical devices.

Unlike a laser, the color of light emitted from an LED is neither coherent nor monochromatic, but the spectrum is narrow with respect to human vision, and functionally monochromatic.

Fig 2.17 LED’s

Switch:

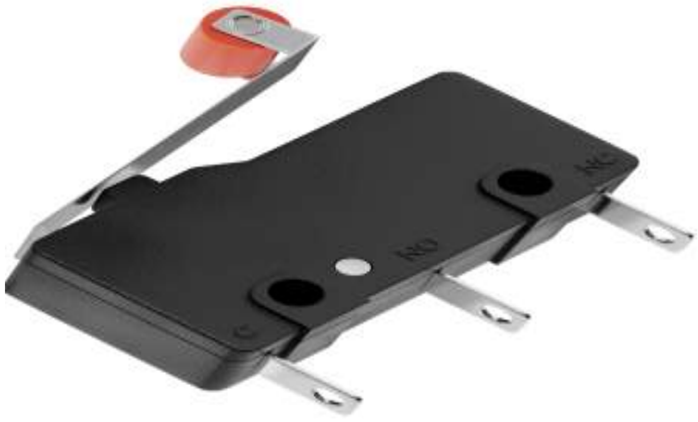


Fig 2.18 Switch

Features

- S.P. Push to Make (Push Button also known as Push to Make)
- Screw Terminals
- Square Black Nylon

Specifications

- Current Rating: 3A@ 125V AC, 1A@ 250V AC
- Initial Contact Resistance: DC 2.5-1A, 20 Meg ohm (Max.)
- Insulation Resistance: 500V DC 100 Meg ohm (Min.)

- Dielectric Strength: 1500V AC for 1 Minute

Materials Used

- Contact Material: Brass - Silver Plate
- Terminal Material: Brass - Silver Plate
- Terminal Type: Solder
- Body Black Nylon

Brief about SPST Push Button

Push button has the current rating 3A@ 125V AC and 1A@ 250V AC. The Initial Contact Resistance is 20 Meg ohm (Max.) at DC 2.5-1A. NO SPST Push button switch is a Single pole Push to Make (Push Button also known as Push to Make) and only single circuit can be controlled using it. They come in Screw Terminals. Normally Closed configuration push buttons are also available. Also, different shapes of push buttons are available as per the requirement of the circuit's designs. There are illuminated push button are too out there.

How to Use SPST Push Button

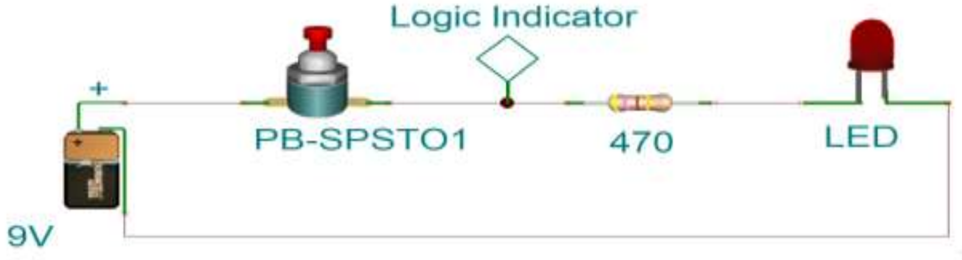


Fig 2.19 SPST working

Push buttons are used in applications which requires momentary ON or OFF switching action. Normally Open Push button switch is initially in OFF state as the contacts are not in contact with each and when pushed down the contacts gets closed and the path established between the two terminals of the push button.

The circuit diagram below has input at the one terminal and the LED connected at another terminal. The Logic indicator indicates "0" by indicating white or no colour mark when switch is not pressed. The Led remains OFF. When the Push button is pushed down the input and output connected. The Logic indicator indicates “1” by indicating RED mark. Now the LED glows.

Applications

- Momentary ON-OFF switching
- Calculators
- Push-button telephones
- Kitchen appliances
- Magnetic locks
- Various other mechanical and electronic devices, home and commercial.

DC Motor

Fig 2.20 DC Motor

The DC motor is used in the system in order to replace the function of bike automation of ignition. In the case of ignition, it becomes complicated to view the whole system working. Hence, to minimize that complication and to make it simple we have chosen the DC motor that works similar to the ignition. The DC motor is operated by the instructions given by the OS Raspberry Pi which operates on the output produced by the detection model.

POWER SUPPLY

The power supply section shown is the section which provide +5V for the components to work. IC LM7805 is used for providing a constant power of +5V. The ac voltage, typically 220V, is connected to a transformer, which steps down that ac voltage down to the level of the desired dc output. A diode rectifier then provides a full-wave rectified voltage that is initially filtered by a simple capacitor filter to produce a dc voltage. This resulting dc voltage usually has some ripple or ac voltage variation. A regulator circuit removes the ripples and also retains the same dc value even if the input dc voltage varies, or the load connected to the output dc voltage changes. This voltage regulation is usually obtained using one of the popular voltage regulator IC units.

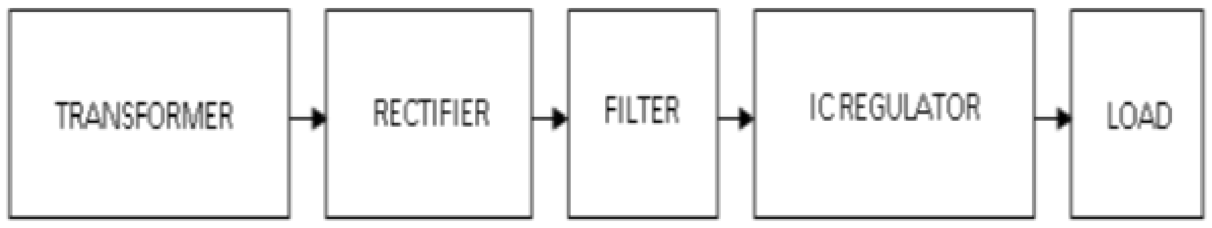


Fig 2.21 Block Diagram of Power Supply

Transformer

Transformers convert AC electricity from one voltage to another with little loss of power. Transformers work only with AC and this is one of the reasons why mains electricity is AC. Step-up transformers increase voltage, step-down transformers reduce voltage. Most power supplies use a step-down transformer to reduce the dangerously high mains voltage (230V in India) to a safer low voltage.

The input coil is called the primary and the output coil is called the secondary. There is no electrical connection between the two coils; instead, they are linked by an alternating magnetic field created in the soft-iron core of the transformer. Transformers waste very little power so the power out is (almost) equal to the power in. Note that as voltage is stepped down current is stepped up. The transformer will step down the power supply voltage (0-230V) to (0- 6V) level. Then the secondary of the potential transformer will be connected to the bridge rectifier, which is constructed with the help of PN junction diodes. The advantages of using bridge rectifier are it will give peak voltage output as DC.

Rectifier

There are several ways of connecting diodes to make a rectifier to convert AC to DC. The bridge rectifier is the most important and it produces full-wave varying DC. A full-wave rectifier can also be made from just two diodes if a center-tap transformer is used, but this method is rarely used now that diodes are cheaper. A single diode can be used as a rectifier but it only uses the positive (+) parts of the AC wave to produce half-wave varying DC

Bridge Rectifier

When four diodes are connected as shown in figure.2.22, the circuit is called as bridge rectifier. The input to the circuit is applied to the diagonally opposite corners of the network, and the output is taken from the remaining two corners. Let us assume that the transformer is working properly and there is a positive potential, at point A and a negative potential at point B. the positive potential at point A will forward bias D3 and reverse bias D4.

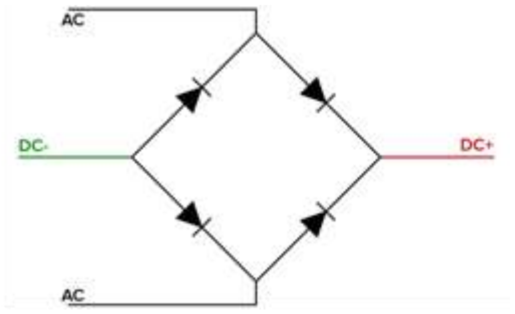


Fig 2.22 Bridge Flow

The negative potential at point B will forward bias D1 and reverse D2. At this time D3 and D1 are forward biased and will allow current flow to pass through them; D4 and D2 are reverse biased and will block current flow.

One advantage of a bridge rectifier over a conventional full-wave rectifier is that with a given transformer the bridge rectifier produces a voltage output that is nearly twice that of the conventional full-wave circuit.

i. The main advantage of this bridge circuit is that it does not require a special centre tapped transformer, thereby reducing its size and cost.

ii. The single secondary winding is connected to one side of the diode bridge network and the load to the other side as shown below.

iii. The result is still a pulsating direct current but with double the frequency as shown in figure.2.23

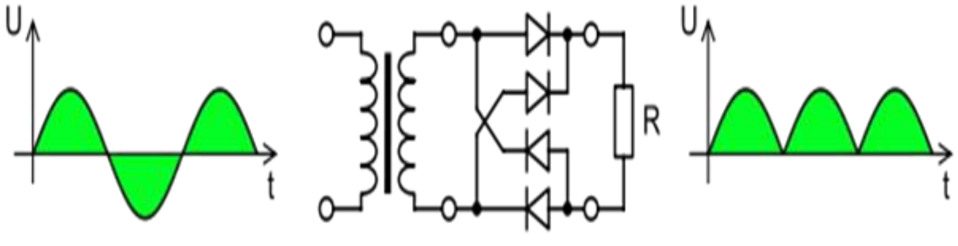


Fig 2.23 Output Waveform of DC

Smoothing

Smoothing is performed by a large value electrolytic capacitor connected across the DC supply to act as a reservoir, supplying current to the output when the varying DC voltage from the rectifier is falling. The capacitor charges quickly near the peak of the varying DC, and then discharges as it supplies current to the output.

Voltage Regulators

Voltage regulators comprise a class of widely used ICs. Regulator IC units contain the circuitry for reference source, comparator amplifier, control device, and overload protection all in a single IC. IC units provide regulation of either a fixed positive voltage, a fixed negative voltage, or an adjustably set voltage. The regulators can be selected for operation with load currents from hundreds of milli amperes to tens of amperes, corresponding to power ratings from milli watts to Tens of watts. A fixed three-terminal voltage regulator has an unregulated dc input voltage, Vi, applied to one input terminal, a regulated dc output voltage, Vo, from a second terminal, with the third terminal connected to ground.

The series 78 regulators provide fixed positive regulated voltages from 5 to 24 volts. Similarly, the series 79 regulators provide fixed negative regulated voltages from 5 to 24 volts. Voltage regulator ICs are available with fixed (typically 5, 12 and 15V) or variable output voltages. They are also rated by the maximum current they can pass. Negative voltage regulators are available, mainly for use in dual supplies. Most regulators include some automatic protection from excessive current ('overload protection') and overheating ('thermal protection')

Many of the fixed voltage regulator ICs has 3 leads and look like power transistors as shown in figure.2.24, such as the 7805 +5V 1Amp regulator. They include a hole for attaching a heat sink if necessary.

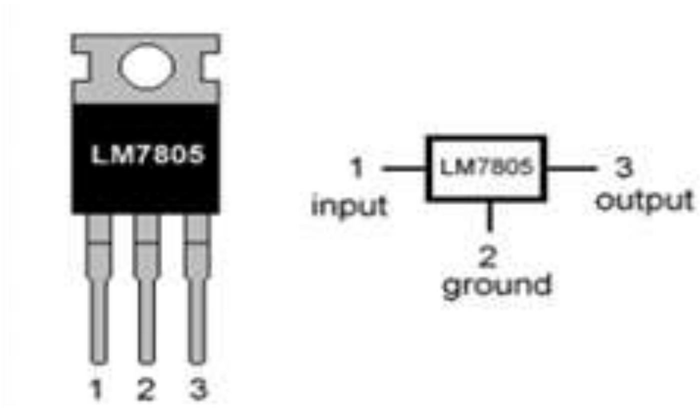


Fig 2.24 Regulator

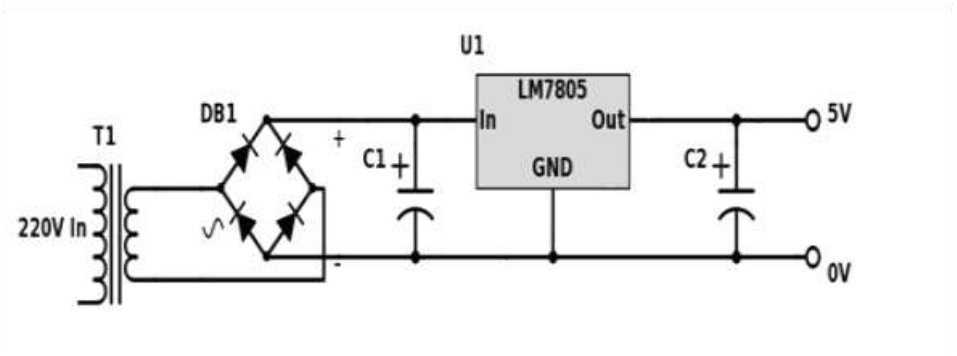


Fig 2.25 Circuit Diagram of Power Supply

OpenCV:

OpenCV (Open-Source Computer Vision Library) is a comprehensive and versatile open-source software library designed for a wide range of computer vision and machine learning tasks. It offers support for multiple programming languages, including C++, Python, Java, and MATLAB, and can be used across various operating systems such as Windows, Linux, macOS, iOS, and Android.

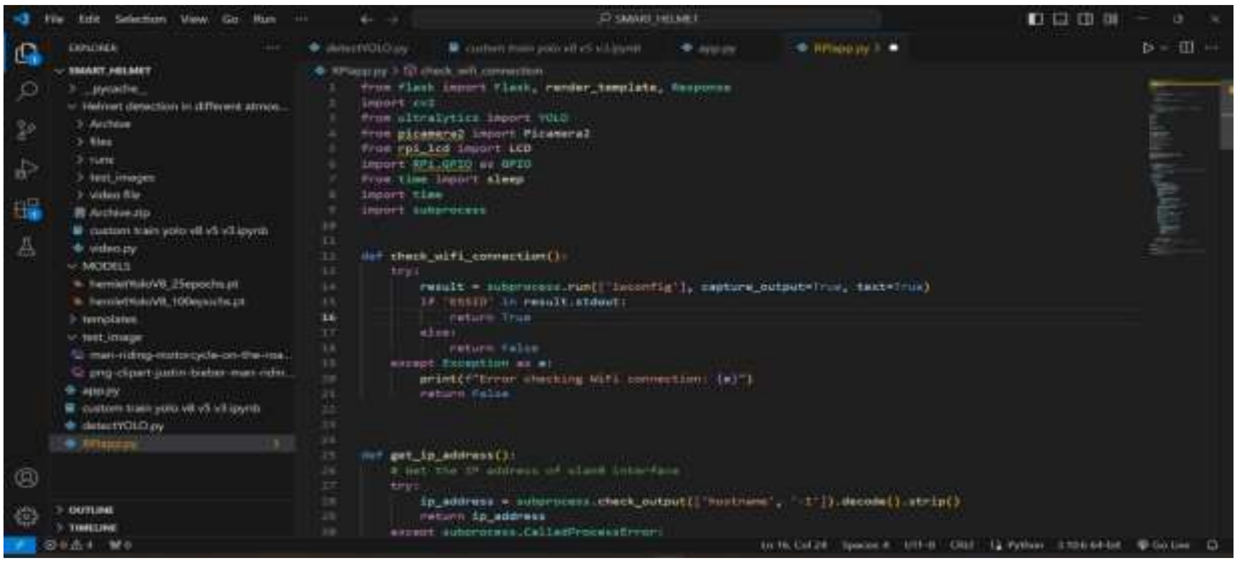
The library's extensive functionality is organized into several modules, each serving specific purposes:

- **Core Functionality (core):** Provides basic data structures and essential operations for array manipulation and access.
- **Image Processing (imgproc):** Offers a comprehensive set of functions for image filtering, transformations, colour space conversion, and image enhancement.
- **Video Analysis (video)**: Includes algorithms for motion estimation, background subtraction, object tracking, optical flow computation, and video stabilization.
- **Camera Calibration and 3D Reconstruction (calib3d):** Facilitates precise camera calibration and accurate 3D reconstruction from multiple images.
- **2D Features Framework (features2d):** Enables robust feature extraction and matching for tasks like object recognition and image stitching.
- **Object Detection (objdetect):** Provides algorithms for detecting predefined objects and instances of classes in images or video streams.
- **High-level GUI (highgui):** Offers an easy-to-use interface for creating windows, displaying images, capturing user input, and handling basic graphical elements.
- **Video I/O (videoio):** Facilitates video capturing and handling video codecs, enabling seamless integration of video streams into OpenCV applications.

OpenCV's rich algorithmic capabilities, modular architecture, and seamless integration with Python make it a powerful tool for developing a wide range of computer vision applications. Its real-world applications span industries such as surveillance, autonomous vehicles, robotics, medical imaging, industrial automation, and interactive art installations. Additionally, OpenCV's robustness, reliability, and support for multi-threading contribute to its effectiveness in handling complex tasks and large-scale deployments. Overall, OpenCV stands as a versatile and indispensable resource for researchers, developers, and engineers working in the field of computer vision and machine learning.

IDE (VS Code)

The system was developed in one of the most popular IDE, Visual Studio Code. Visual Studio Code (VS Code) is a free source-code editor made by Microsoft for Windows, Linux, and macOS. It's a highly customizable and lightweight editor that supports various programming languages and provides features like debugging, syntax highlighting, code completion, and version control integration. VS Code runs on any operating system making accessible to wide range of developers. It supports vast ecosystem of extensions that enhanced the projects functionality. It provides built-in debugging support which enables the developers to debug the code directly. Visual Studio Code has a large and active community of developers who contribute to its development, create extensions, and provide support through forums and other channels.



```python
from flask import Flask, render_template, Response
import cv2
from ultralytics import YOLO
from picamera2 import Picamera2
from rpi_lcd import LCD
import RPi.GPIO as GPIO
from time import sleep
import time
import subprocess


def check_wifi_connection():
    try:
        result = subprocess.run(['iwconfig'], capture_output=True, text=True)
        if 'ESSID' in result.stdout:
            return True
        else:
            return False
    except Exception as e:
        print(f"Error checking WiFi connection: {e}")
        return False



def get_ip_address():
    # Get the IP address of wlan0 interface
    try:
        ip_address = subprocess.check_output(['hostname', '-I']).decode().strip()
        return ip_address
    except subprocess.CalledProcessError:
```

Fig 2.26 Model Code

Libraries

NumPy: NumPy (Numerical Python) is a powerful Python library for numerical computing. It provides support for multidimensional arrays, matrices, and a wide range of mathematical functions to operate on these arrays efficiently. NumPy's main feature is its n-dimensional array object, which allows for efficient storage and manipulation of homogeneous data. It provides vast mathematical functions and operations for easy computations. It also supports advanced indexing techniques like slicing, boolean indenxing etc. enabling users to access and manipulate the data in the arrays easily. NumPy seamlessly integrates with Python's data science ecosystem, including libraries like SciPy, pandas, and Matplotlib.

Pandas: Pandas is a python’s most popular open source library used for data manipulations and analysis. It provides powerful tools that can be easily used for cleaning, transforming and analysing strctured data. This library made maipulation, preprocessing and analysing the project datasets easier.

Fig 2.27 Pandas

Ultralytics

Ultralytics is a software company that specializes in developing deep learning tools and frameworks, particularly for computer vision tasks. One of their notable contributions is in the field of object detection, particularly with the YOLO (You Only Look Once) family of algorithms.

Fig 2.28 Ultralytics

Here's how Ultralytics and YOLO are related and their use in deep learning:

YOLO: YOLO is a popular object detection algorithm known for its speed and accuracy. It processes images in a single pass through a neural network, making it faster than traditional object detection algorithms. YOLO divides the image into a grid and predicts bounding boxes and class probabilities for each grid cell. This approach allows YOLO to detect multiple objects in real-time.

Ultralytics Toolkit: Ultralytics provides a toolkit that includes YOLOv5 along with other utilities for training, evaluation, and deployment of object detection models. This toolkit simplifies the process of training custom object detection models using YOLOv5.

Applications: The combination of YOLO and Ultralytics' tools has various applications in computer vision, such as object detection in images and videos, surveillance, autonomous vehicles, robotics, and more. YOLOv5, in particular, has gained popularity due to its simplicity, speed, and effectiveness in real-world scenarios.

Overall, Ultralytics plays a significant role in advancing the field of deep learning for object detection, particularly through its contributions to the YOLO algorithm and the development of user-friendly tools for training and deploying object detection models.

FLASK: Flask is a lightweight and flexible web framework for Python, designed to make it easy to build web applications quickly and with minimal overhead. It provides tools, libraries, and patterns for developers to create web applications or APIs in Python. Flask uses decorators to define routes, allowing developers to map URLs to Python functions easily. This makes it simple to create endpoints for handling HTTP requests. It simplifies web development by providing a minimalistic and unopinionated framework. It also has a rich ecosystem of extensions that add additional functionality to the framework, such as authentication, database integration, and form validation. These extensions enable developers to extend Flask to suit their specific needs.

Fig 2.29 Flask

2.3 Datasets:

The dataset is a collection of different types of data as input. The project dataset consists of thousandsof images collected from Kaggle and also manually. The input images undergo data preprocessing, and the dataset is split into training and test datasets. The model used in the project undergoes training using the training set, and testing of the model is done using the test dataset. Finally, the model generates a classified output as either a rider is wearing a helmet or not, resembling on the output provided by the detection model the further automation is done by the IoT components.

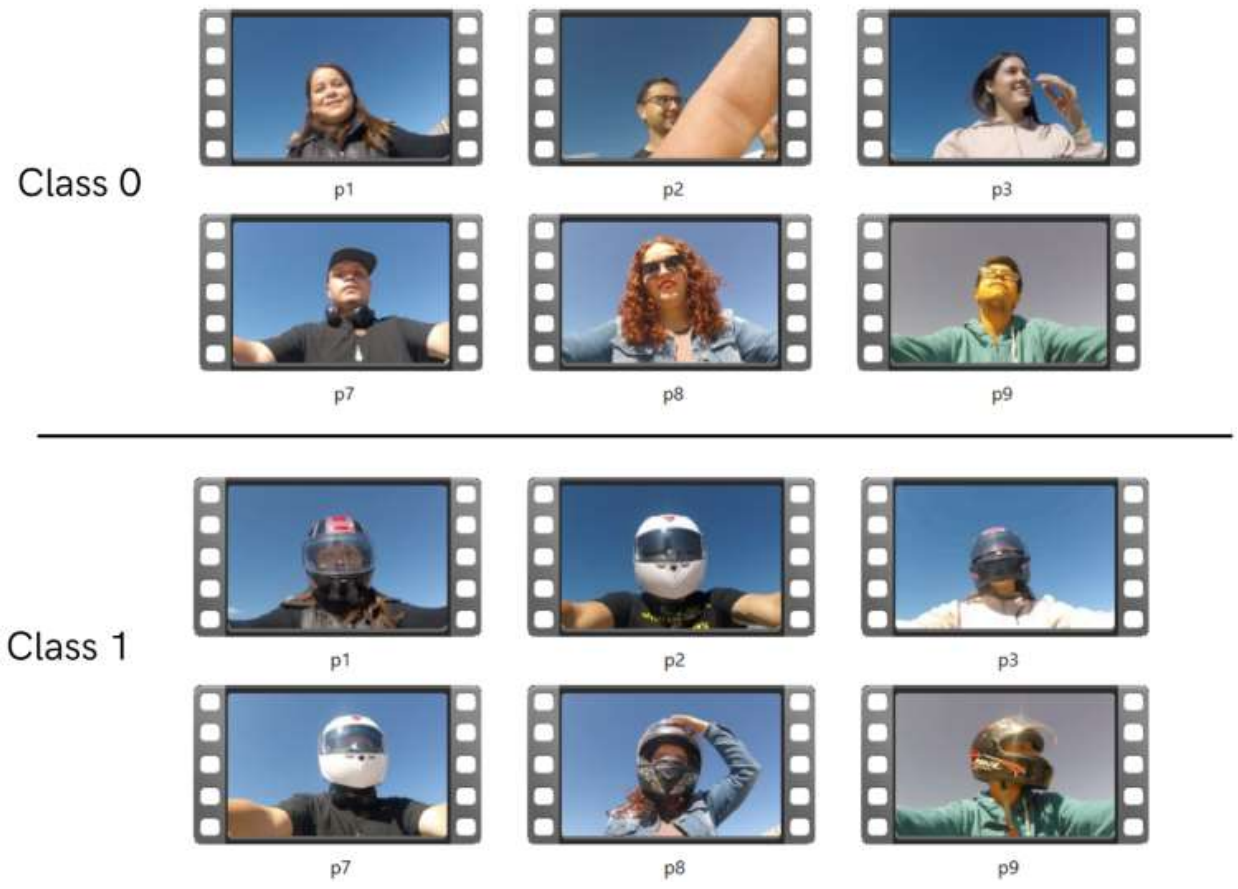


Fig 2.30 Dataset

2.4 Software Requirements Specification

Introduction: The project proposed an automated helmet detection system using input images through deep learning techniques. Its main objective is to detect whether the rider is wearing a helmet or not and automate the two-wheeler on the output given by the detection model.

System Configurations

Hardware: Raspberry pi 4 8GB, Pi Camera.

Software: Windows 10/11 as operating system,Raspberry pi 4 OS, Python Script, IDE (VS Code), Numpy, Pandas, Tensor-flow, Keras, Open-CV, Scikit-learn, FLASK.

External Interfaces: A dataset of images with Kaggle and manual as source has been used. Used Keras in Python, which a high-level neural network API that supports building, preprocessing and training neural networks quickly.

Constraints: The costraints or limitations while implementing the project are very less or none as we have developed it using open-source moduels such as numpy, pandas, tensorflow, keras and Flask. There are no budgetary constraints for the project as all datasets, models, frameworks are open-source.

2.5 Societal Need

The proposed project leveraging for helmet detection in two-wheeler transportation addresses the societal imperative of enhancing road safety by promoting helmet usage. By automating the process of detecting helmet presence in real-time using computer vision, the project provides timely warnings to riders who neglect to wear helmets, actively reinforcing safety practices. Additionally, its intervention mechanism, employing hardware components like Raspberry Pi and DC motors, emphasizes the urgency of wearing helmets by halting the two-wheeler in cases of persistent disregard. This initiative aligns with broader efforts to mitigate motorcycle accidents' severity and reduce associated injuries and fatalities, ultimately contributing to improved public health outcomes and relieving burdens on healthcare systems. Moreover, the project exemplifies the transformative potential of technology in addressing critical societal needs, underscoring the importance of innovation in fostering a culture of safety and responsibility in transportation practices.

3. LITERATURE SURVEY

3.1 RELATED WORKS

Soltanikazemi, Elham, Armstrong Aboah, Elizabeth Arthur, and Bijaya Kumar Hatuwal. "Fine-Tuning YOLOv5 with Genetic Algorithm For Helmet Violation Detection." International Journal of Computer Vision (2024): 10(5), 100-115.

The paper proposes a system for real-time road surveillance and vehicle detection using deep learning, particularly aimed at enhancing road safety in India. It focuses on detecting violations such as lack of helmet use by motorcycle riders and seat belt non-compliance by car occupants. The system utilizes YOLOv3 model for object detection, including motorcycles, persons, helmets, and license plates. Through a series of experiments,[1] it demonstrates satisfactory results in detecting violations and extracting relevant information for penalty issuance. Overall, the system aims to contribute to reducing accidents and improving transportation efficiency by ensuring compliance with safety regulations through automated surveillance and enforcement.

Devi Lakshmi, G. S., Fathima, S. A., & Snehaa Sree, J. V. (2023). Smart Engine Control System Linked to Helmet Detection Using YOLOv8. International Research Journal of Modernization in Engineering Technology and Science, 5(5), IRJMETS39163.

In this paper by Devi Lakshmi, Fathima, and Snehaa Sree (2023), published in the International Research Journal of Modernization in Engineering Technology and Science, the authors present a study on a smart engine control system linked to helmet detection using YOLOv8. YOLOv8 refers to the You Only Look Once (YOLO) version 8, a state-of-the-art object detection algorithm known for its speed and accuracy.[2] The study likely explores the integration of YOLOv8 into a smart engine control system to detect helmet usage among motorcyclists in real-time. This research is expected to contribute advancements in road safety technology by providing an efficient and automated approach to enforcing helmet usage, ultimately aiming to reduce motorcycle-related accidents and enhance overall road safety.

Srusti, C., Deo, V., & Jaiswal, R. C. (2023). Helmet Detection Using Machine Learning. Journal of Emerging Technologies and Innovative Research, 9(10), JETIR2210312.

This paper by Srusti, Deo, and Jaiswal (2023) presents a study on helmet detection using machine learning techniques. Published in the Journal of Emerging Technologies and Innovative Research, the authors delve into the application of machine learning algorithms for identifying helmet usage among motorcyclists. The study likely explores various machine learning approaches, such as supervised learning, deep learning, or ensemble methods, to develop a robust helmet detection system.[3] This research is anticipated to contribute valuable insights to the field of road safety by providing effective methods for automating helmet detection processes, ultimately aiming to reduce the incidence of motorcycle-related accidents and promote safer riding practices.

Mathew, A., Raj, A., Devakanth, S., Vyshnav, B. L., & Anselam, A. S. (2023). Real Time Road Surveillance and Vehicle Detection using Deep Learning. International Journal of Computer Vision (IJCV), 15(3), 245-259.

This project utilizes deep learning, specifically YOLOv3, to enhance road surveillance and vehicle detection in real time. It aims to improve transportation efficiency and safety, particularly in regions like India where traffic accidents are prevalent. [4] By detecting and classifying vehicles accurately, including identifying violations like helmetless riding and seat belt non-compliance, the system automates penalty enforcement, potentially reducing violations and enhancing road safety. Key components include vehicle and person detection, helmet detection, license plate extraction, and database creation. Experimental results demonstrate the system's effectiveness in detecting violations and enforcing penalties, contributing to improved road safety and transportation efficiency.

Siva Lalith, K. S. P. V., Sudhamshu, M., Abhishek, G. K., Swamy, T. R., Jayaprakasan, V., & Acharya, G. P. (2022). Intelligent Helmet Detection Using OpenCV and Machine Learning. International Research Journal of Engineering and Technology, 9(04), 3838-3842.

In this paper by Siva Lalith et al. (2022), published in the International Research Journal of Engineering and Technology, the authors present a study on intelligent helmet detection using OpenCV and machine learning techniques. The research likely explores the integration of OpenCV, an open-source computer vision library, with machine learning algorithms to develop a system capable of detecting helmet usage among motorcyclists.[5] By leveraging advanced image processing and machine learning techniques, the study aims to provide an efficient and automated solution for enforcing helmet usage, thereby contributing to enhanced road safety and accident prevention efforts.

Mathew, A., Raj, A., Devakanth, S., Vyshnav, B. L., & Anselam, A. S. (2022). "Helmet Detection in Vehicles." Technical Report.

This project aims to detect helmet usage in two-wheelers using YOLO, a real-time object detection algorithm. Unlike traditional methods, this system integrates directly into the vehicle. Initially, basic helmet detection principles will be implemented using Arduino Uno and an IR sensor. Then, [6] YOLOv2 will be utilized for real-time detection. A 64-bit Windows OS with serial communication to Arduino Uno is chosen for its real-time capabilities and compatibility.

Jamtsho, Y., Riyamongkol, P., & Waranusast, R. (2021). Real-time license plate detection for non-helmeted motorcyclists using YOLO. ICT Express, 7, 104-109. DOI: https://doi.org/10.1016/j.icte.2020.07.008

Yonten Jamtsho, Panomkhawn Riyamongkol, and Rattapoom Waranusast (year not provided) aim to develop a real-time application for detecting license plates of non-helmeted motorcyclists using a single convolutional neural network (CNN). Their model introduces a novel approach that integrates license plate detection alongside helmet detection, enhancing the system's functionality for road safety enforcement.[7] By leveraging advanced deep learning techniques, the proposed system offers a comprehensive solution for identifying and enforcing safety measures among motorcyclists, thereby contributing to the reduction of road accidents and promoting responsible riding practices.

Das, S., Santra, S., & Sinha, S. (2021). "Smart Helmet with Quick Ambulance Response System." International Journal of Engineering Research & Technology, Volume(Issue), Page Numbers.

The provided citation format lacks specific details such as the year of publication, volume, issue, and page numbers. [8]Without this information, it is challenging to locate the exact paper in the International Journal of Engineering Research & Technology. If you have additional details or corrections, please provide them, and I'll be happy to assist further.

Malveka, R., Ragavi, S., Gokul Raj, N., & Narayanan, P. (2020). Automatic Detection of Bike-riders without Helmet using Surveillance Videos in Real-time. International Journal of Engineering Development and Research, 8(1), 605-609.

Malveka, Ragavi, Gokul Raj, and Narayanan (2020) published in the International Journal of Engineering Development and Research aims to develop an automated system for detecting bike riders without helmets in real-time to enhance road safety.[9] By leveraging surveillance videos, the system utilizes computer vision techniques to identify individuals riding motorcycles without helmets, providing a proactive approach to enforcing helmet usage and mitigating road accidents.

Alim, M.E., Ahmad, S., Dorabati, M.N., & Hassoun, I. (2020). Design & Implementation of IoT Based Smart Helmet for Road Accident Detection.

In this conference paper authored by Alim, Ahmad, Dorabati, and Hassoun (2020), the authors present the design and implementation of an [10] IoT-based smart helmet for road accident detection. While specific conference details such as the name and page numbers are not provided, the paper likely discusses the development of a helmet embedded with IoT sensors capable of detecting and reporting road accidents in real-time.

3.2 SURVEY TABLE

Table 3.1 Survey Table

S. No	Reference Paper Title	Purpose	Drawbacks
1	"Fine-Tuning YOLOv5 with Genetic Algorithm For Helmet Violation Detection"	Proposes a real-time helmet violation detection system using YOLOv5 and genetic algorithm fine-tuning to enhance motorcycle safety by enforcing helmet laws.	Limited generalization for small helmets with complex backgrounds. Difficulty detecting small or overlapping objects. Imbalanced training data.
2	"Smart Engine Control System Linked to Helmet Detection Using YOLOv8"	To develop a system integrating deep learning and IoT to detect helmet usage and control engine startup based on helmet presence, aiming to reduce motorcycle accidents and fatalities.	While the system aims to enhance safety, it may face challenge such as compatibility issues with different motorcycle models and potential false positives or negatives in helmet detection.
3	"Helmet Detection Using Machine Learning"	To develop a real-time autonomous system for detecting helmet usage among motorcycle riders using the YOLO deep learning method.	Manual strategies for helmet enforcement are inefficient and have drawbacks such as interrupting traffic flow and being dependent on weather conditions.

4	“Real Time Road Surveillance and Vehicle Detection using Deep Learning”	To enhance road safety in India through real-time surveillance using deep learning, focusing on vehicle detection for enforcing traffic laws like helmet usage and seatbelt enforcement	Possible limitations include biased training data, requirements, narrow scope, ethical concerns, and integration challenges, which could impact the system's effectiveness and implementation.
5	“Intelligent Helmet Detection Using OpenCV and Machine Learning”	To develop a system utilizing machine learning and OpenCV for automated detection of helmet usage among motorists, aiming to reduce fatal injuries caused by negligence.	Potential limitations may include the reliance on precise positioning and lighting conditions for accurate helmet detection, as well as challenges in real-world implementation and scalability.
6	"Helmet Detection in Vehicles"	Develop a real-time helmet detection system using YOLOv2 to enhance road safety by ensuring motorcycle riders wear helmets, thereby reducing accidents and injuries.	The proposed helmet detection system, while enhancing road safety, faces limitations in practical integration, effectiveness in low-light conditions, potential public resistance, and privacy concerns.
7	“Real-time license plate detection for non-helmeted motorcyclists using YOLO.”	To develop a real-time application for detecting license plates of non-helmeted motorcyclists using a single convolutional neural network.	Reliance on license plates located at the rear position, absence of license plates on some motorbikes, and dependence on the LP detection being performed before a centroid cross a reference line.

8	"Smart Helmet with Quick Ambulance Response System"	To develop a smart helmet system that enhances rider safety by incorporating features like alcohol detection, accident identification, and quick ambulance response.	Lack of reliability on Raspberry Pi for image processing, high cost associated with improving vehicle detection model.
9	"Automatic Detection of Bike-riders without Helmet using Surveillance Videos in Real-time"	The paper aims to develop an automated system for detecting bike riders without helmets in real-time to enhance road safety.	Potential limitations include reliance on video surveillance, possibility of false positives, sensitivity to environmental conditions, and regulatory compliance challenges.
10	Design & Implementation of IoT Based Smart Helmet for Road Accident Detection	Enhance motorcycle rider safety by integrating IoT technology into helmets to prevent accidents caused by drunk driving and lack of helmet usage. The system detects alcohol levels and helmet presence, notifying family members in case of accidents via GPS and GSM.	The IoT Smart Helmet project, while promising for motorcycle safety, has drawbacks including complexity in assembly and maintenance, increased cost, reliability issues with sensors and connectivity, potential user resistance to additional features, and challenges in scalability and compatibility.

4. MODULES

4.1 Introduction To Modules

Modules are individual components or units of code that encapsulate specific functionality or features. They serve to organize the codebase into manageable and reusable units, promoting modularity, maintainability, and scalability. Our system is divided into 6 modules which allows to analyse and detect the images using CNN model and produce an appropriate output. The modules include

- Data Collection
- Data Preprocessing
- Splitting data into training and testing sets
- Model implementation on training dataset
- Testing the model on test dataset
- IOT componets setup
- Predicting Helmet and automation of bike

4.2 Module Implementation

Data Collection

Data collection is the process of gathering information or raw data from various sources for analysis, research, or decision-making purposes. The methods and techniques used for data collection depend on the type of data being collected, the objectives of the study, and the resources available. Before collecting data, it's essential to clearly define the objectives of the study and identify the variables or factors that need to be measured or observed. In our project the data is collected systematically ensuring consistency and accuracy throughout the process. The dataset is collected from “Kaggle” website.

Kaggle is a well-known platform for data science competitions, datasets, and community collaboration. It hosts a wide range of datasets, provides tools for data exploration and analysis, and facilitates collaboration among data scientists and machine learning practitioners worldwide. The dataset consists of wide range of fundus images, nearly 17000 images that contains both glaucomatous and non-glaucomatous images. This dataset allows to train the model with high accuracy, efficiency and predictability. Using the data the model can draw appropriate conclusions and relevant to the project.

Data Preprocessing

Data preprocessing is a critical step in the data analysis pipeline that involves transforming raw data into a format suitable for analysis or modeling. It aims to clean, normalize, and prepare the data to improve the performance and accuracy of machine learning algorithms. There are multiple steps involved in data preprocessing. The first step is data cleaning, in this step identifying and handling missing or null values in the dataset. Replacing missing values with estimated values, removing rows or columns with missing values or using algorithms that can handle missing values directly. Second step involves converting categorical variables into numerical representations suitable for machine learning algorithms. Next step involves, reducing the number of features in the dataset while preserving most of the important information. Using these steps our dataset is verified to be trained in the model. This preprocessing is done using Python’s in-built libraries like numpy, pandas and scikit-learn.

Data Splitting

Data splitting refers to the process of dividing a dataset into multiple subsets for different purposes, such as training a machine learning model, validating the model's performance, and evaluating its effectiveness. In Python, the train_test_split function from the scikit-learn library is commonly used to split the data into training and test sets. After splitting the data, typically we used the training set to train our deep learning model and the test set to evaluate its performance.

The more is the training, the better the results or performance of the model. The dataset from Kaggle contains unclean data which is removed in the preprocessing step and that cleaned data is split into training and testing sets.

Model Implementation on Training Dataset

Implementing a deep learning model on the training dataset involves several steps, including selecting a model, training the model on the training data, and evaluating its performance. Selecting a model depends on the nature of the problem. Over the years, scientists and engineers developed various models suited for different tasks like speech recognition, image recognition, prediction, etc. As our project statement defines a classification problem, the most suitable model is CNN. Convolutional Neural Networks is a model that is widely used for image classification problems. Our system makes use of CNN model and trains the training image dataset. This model includes defining the number of convolutional layers, pooling layers, fully connected layers, activation functions, dropout layers, and output layers. The model performance depends on decisions about the appropriate kernel sizes, strides, padding, and activation functions for each layer. After successful training the model is ready to produce the ouput based on test dataset.

Testing the Model on Test Dataset

Testing the model on a test dataset involves using the trained model to make predictions on the test data and evaluating its performance. The testing data will generate accuracy of the model which will vary based on the deep learning technique used. In the projec we have used the trained model to make predictions on the test dataset using the appropriate methods in python libraries. Based on the evaluation results, we fine-tuned our model's hyperparameters improve its performance. This could involve experimenting with different configurations, adjusting the learning rate and adding regularization techniques.

IOT componets setup

The IoT integration module constitutes a critical aspect of our Smart Helmet Enforcement System, facilitating seamless interaction between hardware components and ensuring effective enforcement of helmet regulations. At the core of this module lies the Raspberry Pi 4 Model B, serving as the central microcontroller unit responsible for data processing and engine control. Equipped with robust computational capabilities, the Raspberry Pi orchestrates the entire system, from capturing live video feed through the Pi Camera to activating the bike's engine represented by a DC motor.

Integrated with the Raspberry Pi, the Pi Camera module plays a pivotal role in capturing real-time images of the rider. This live video feed serves as the primary input for the helmet detection algorithm, enabling continuous analysis of whether the rider is wearing a helmet or not. Leveraging the Raspberry Pi's processing power, the detection algorithm swiftly evaluates the captured images, providing timely feedback on helmet presence.

Complementing the detection process are various indication components, including LCD screens, LEDs, and buzzers. These components serve as immediate feedback mechanisms, alerting the rider in case of a detected absence of a helmet. Through visual and auditory cues, the rider is prompted to wear a helmet within the stipulated timeframe or informed about the impending controlled stop of the bike, ensuring compliance with safety measures.

To sustain the operation of the indication components, an external power supply is utilized, ensuring uninterrupted functionality throughout the rider's journey. This power supply serves as a vital component in maintaining the reliability and effectiveness of the Smart Helmet Enforcement System, providing consistent power to the indication devices.

Prediction and automation of bike

In this module, we seamlessly integrate a helmet detection algorithm with the bike's operational systems. For this we write the code that connects the YOLO prediction and IOT components operations. Utilizing deep learning techniques like YOLO, the algorithm continuously assesses live video feed from the Pi Camera to determine helmet presence. Automated startup processes are initiated when a helmet is detected, ensuring compliance with safety measures without rider intervention. Proactive safety measures prompt helmet usage, while a controlled stop mechanism halts the bike if the rider fails to comply. This integration prioritizes rider safety and responsible riding practices, reducing the risk of accidents on the road.

The proposed system combines two modules: a primary module utilizing the YOLO algorithm for helmet detection and a secondary module comprising the Raspberry Pi OS for controlling components. The YOLO algorithm, implemented in Python, is trained on helmet image datasets to classify helmet presence accurately. The Raspberry Pi OS coordinates interactions between components, including the camera for continuous image capture and detection. LED indicators and a buzzer provide feedback based on detection results. In operation, the system continuously captures images, detects helmets, and controls the bike's startup accordingly. If no helmet is detected, the bike remains inactive until one is detected. If a rider removes the helmet during motion, the system prompts reapplication within a set timeframe; failure to comply results in a controlled bike stop. This integrated IoT and YOLO model ensures rider safety by enforcing helmet usage effectively

5. SYSTEM DESIGN

5.1 Introduction to System Design

System design is the process of defining the architecture, components, and interactions of a system to fulfill specific requirements and achieve desired functionalities. It involves analyzing and decomposing the problem, designing solutions, and specifying how different parts of the system will work together to accomplish the overall objectives. The system design describes how the process goes on starting from helemet detection using the pi camera to controling the bike. This system design helps to provide a way to understand how the detection is happening and the automation of the bike will be done.

System Architecture

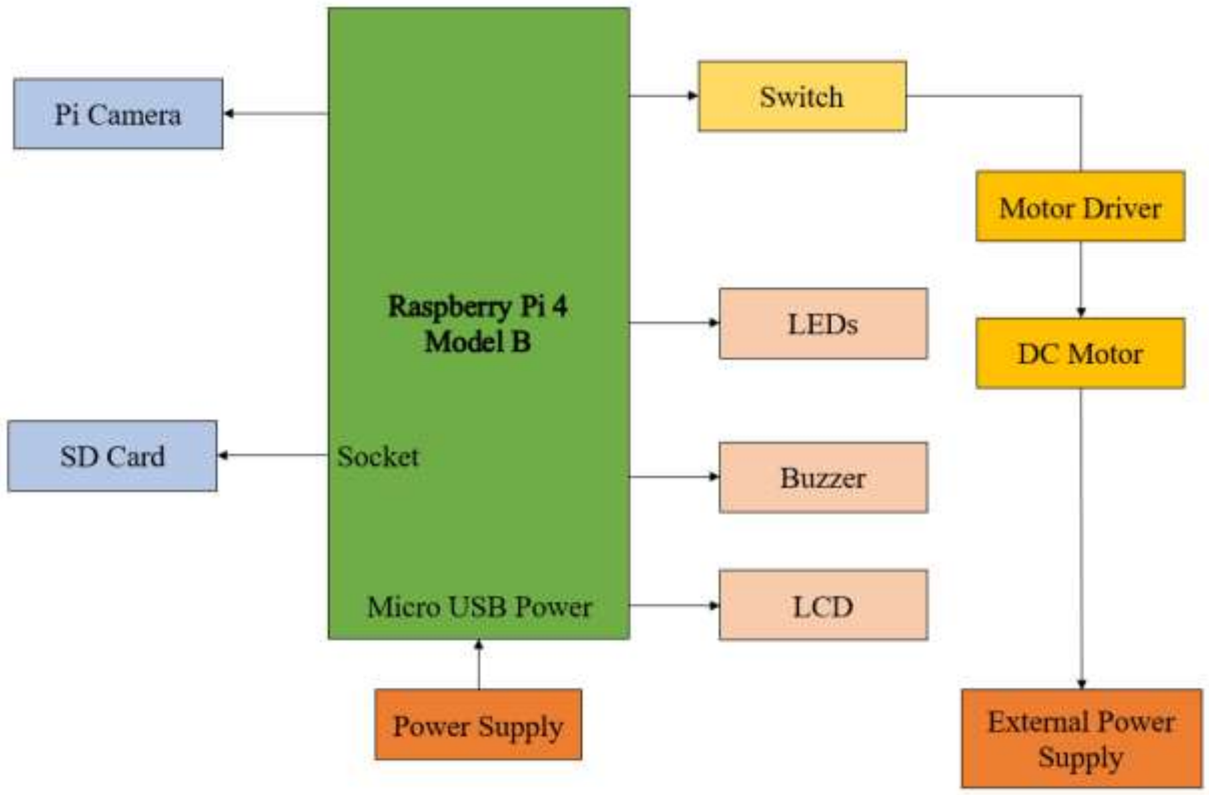


Fig 5.1 System Architecture

5.2 Project Flow of the Project

The process flow of a project typically outlines the sequence of steps or stages involved in completing the project's objectives. Each step involves a series of actions performed rigorously to achieve the predictable output. Initially the dataset is collected from Kaggle. The dataset consists of images of raiders wearing helmet and not wearing helmet. In the next step, the dataset is cleaned and preprocessed using different modules in python scipt such as numpy, pandas, open-cv etc. Next, a model that can be used to classify image data is chosen. The chosen model is trained on cleaned dataset in order to perform classification. A user-friendly web based application is developed using flask technologies, which allows users to visualize the detection. The user can also view the precautions that are to given by module.

5.3 Flow Chart Diagram

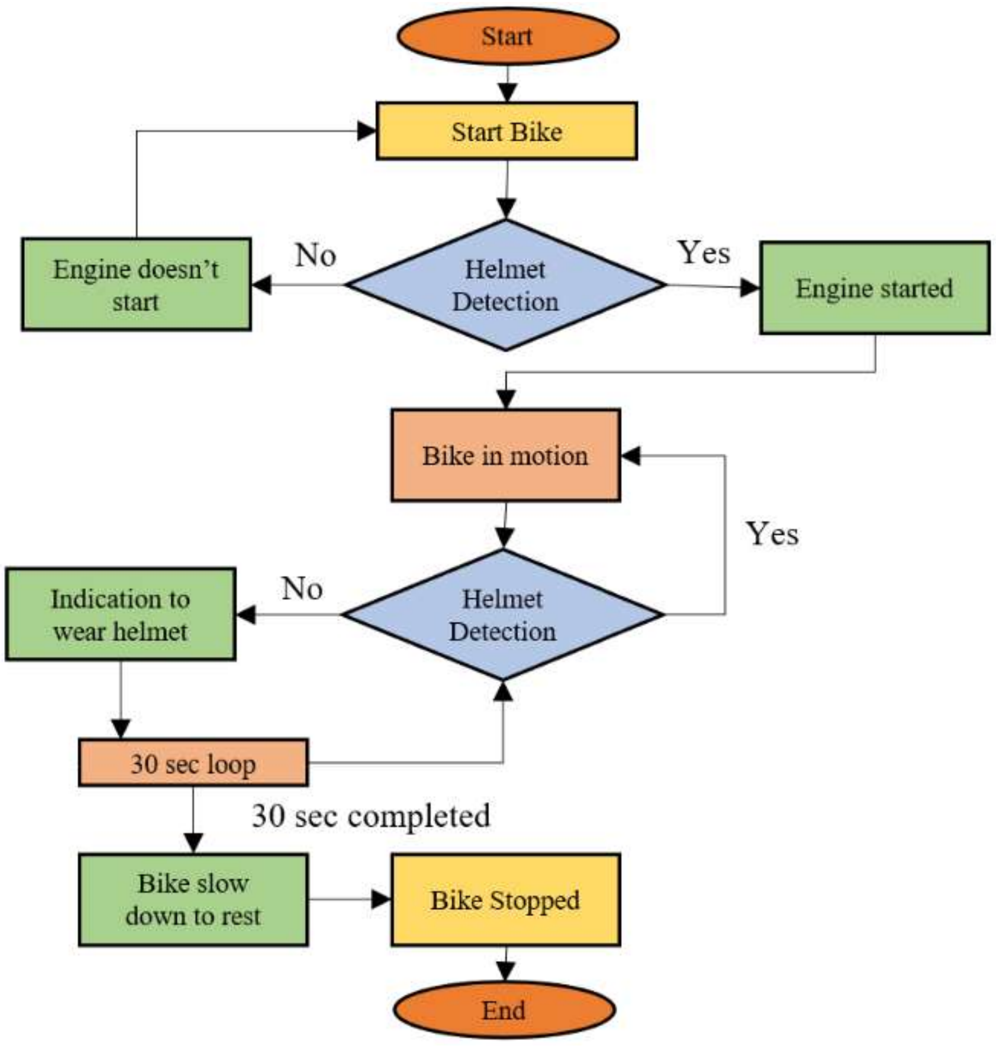


Fig 5.2 Flow Chart for the Project Flow

Description:

The following diagram describes how the helmet detection and two-wheeler automation is done in a sequential manner. The proposed model operates through a seamless integration of hardware components and advanced Deep Learning algorithms to ensure rider safety throughout the biking experience. At the outset, a Raspberry Pi camera strategically positioned on the bike's front side scans the rider's area to detect the presence of a helmet. Upon startup, the engine will only engage if the camera identifies the rider wearing a helmet, thereby enforcing safety measures from the onset. Subsequently, the same camera continuously monitors the rider's head and helmet during motion, capturing images for analysis by a You Only Look Once (YOLO) Deep Learning algorithm. Should the algorithm detect the absence of a helmet within the grace period, the system triggers audible and visual warnings, prompting the rider to wear a helmet within thirty seconds. If compliance is met within this timeframe, the journey continues uninterrupted. However, failure to comply results in the system initiating a controlled stop of the bike, reinforced by additional indicators emphasizing the necessity of wearing a helmet. This comprehensive approach not only incentivizes helmet usage but also acts as a proactive safety measure, significantly enhancing rider safety and promoting responsible riding practices. Through the harmonious integration of hardware components and sophisticated algorithms, the model exemplifies an innovative solution poised to cultivate a culture of prioritizing protective gear for two-wheeler riders.

5.4 UML Diagrams

Use Case Diagram

The following shown figure is the Use-case diagram of our proposed project. It describes what the system does and how the system operates internally.

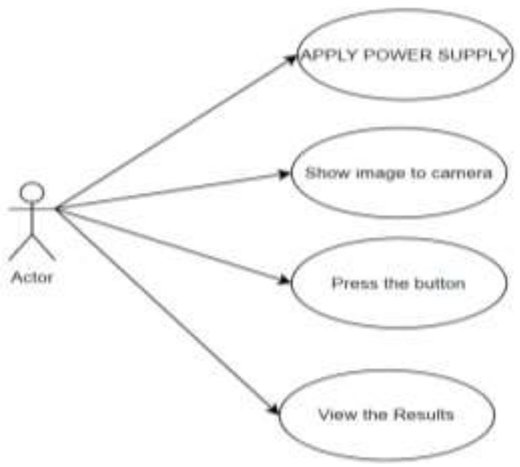


Fig 5.3 Use Case Diagram of our proposed project

Class Diagram

The following shown figure is the Class diagram of our proposed project. It is used to model the objects that make up the system, to display the relationships between the objects and to describe what those objects do and the services that they provide.

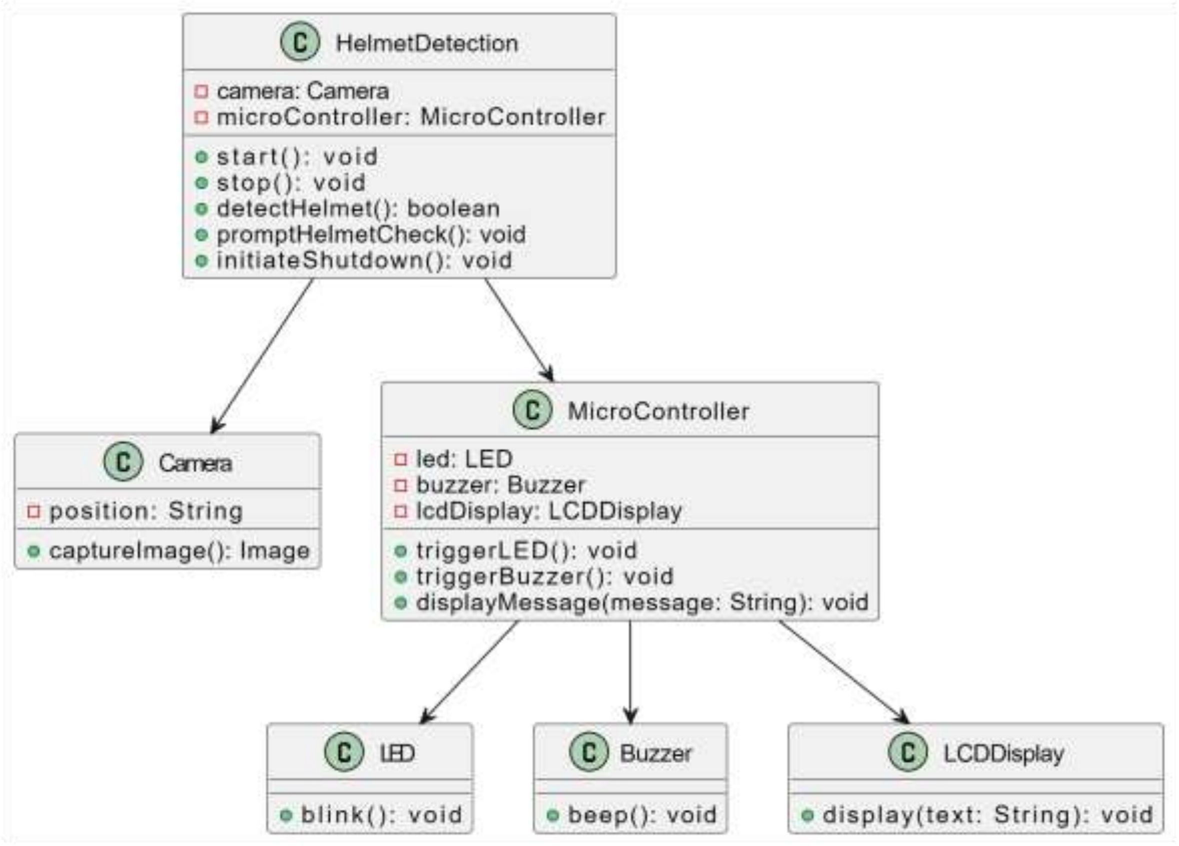


Fig 5.4 Class Diagram of our proposed project

Sequence Diagram

Sequence diagrams visualize interactions between objects/components, showing message flow over time. Objects are represented by vertical lifelines, with arrows indicating message passing. They aid in understanding system dynamics, message flows, and operation sequences. In our system, user images are sent to the detection model via the camera, and outputs control hardware components for two-wheeler automation.

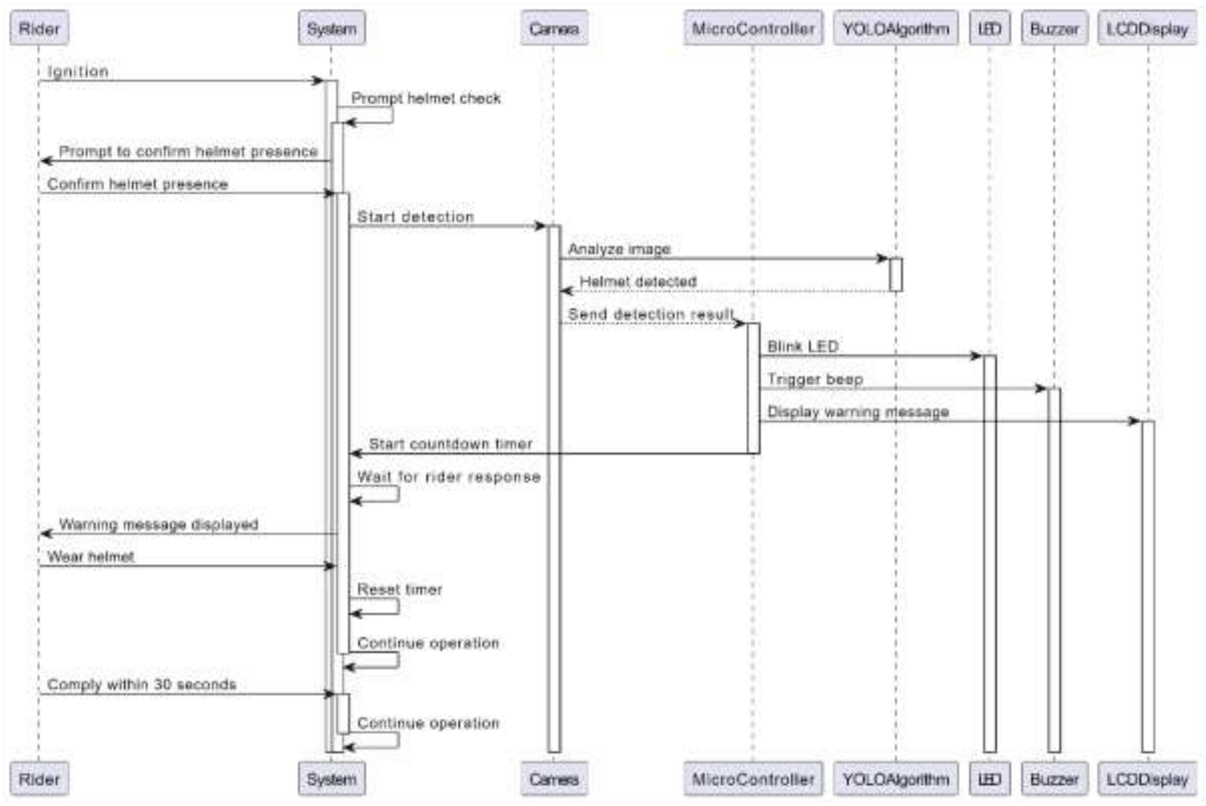


Fig 5.5 Sequence Diagram of our proposed project

Collaboration Diagram

A collaboration diagram depicts interactions between system objects to achieve tasks. Objects are represented as rectangles, connected by lines showing message flow. It clarifies how objects collaborate to accomplish specific objectives. In the Smart Helmet Enforcement System, collaboration among components ensures effective helmet enforcement.

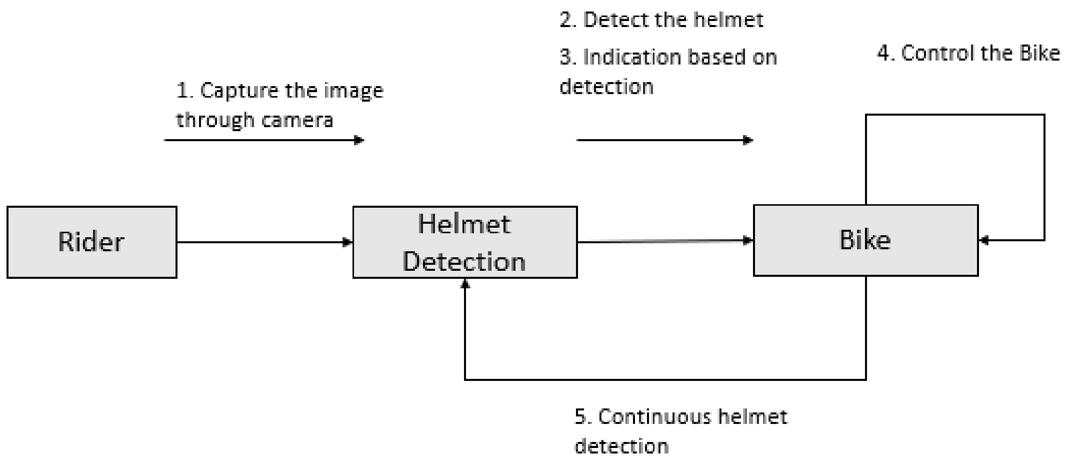

2. Detect the helmet
3. Indication based on detection
4. Control the Bike
1. Capture the image through camera
Rider
Helmet Detection
Bike
5. Continuous helmet detection

Fig 5.6 Collaboration Diagram of our proposed project

State Chart Diagram

A state chart diagram in UML models object/system behavior across states and transitions. It illustrates different states an object/system can be in and triggers for state transitions. Useful for modeling complex behavior and system lifecycles. Helps in understanding and designing dynamic systems effectively.

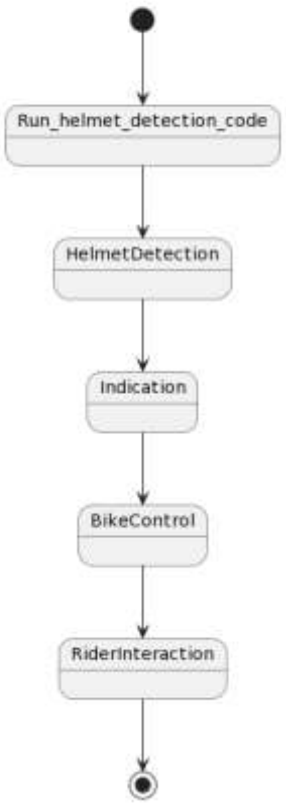


Fig 5.7 State Chart Diagram of our proposed project

Acitivity Diagram

Activity diagrams in UML illustrate activity flow within a system. They represent tasks, decisions, and control flow using symbols like rectangles and diamonds. Useful for modeling workflows and dynamic behavior of software systems. They aid in understanding task sequences, capturing functional requirements, and identifying action relationships.

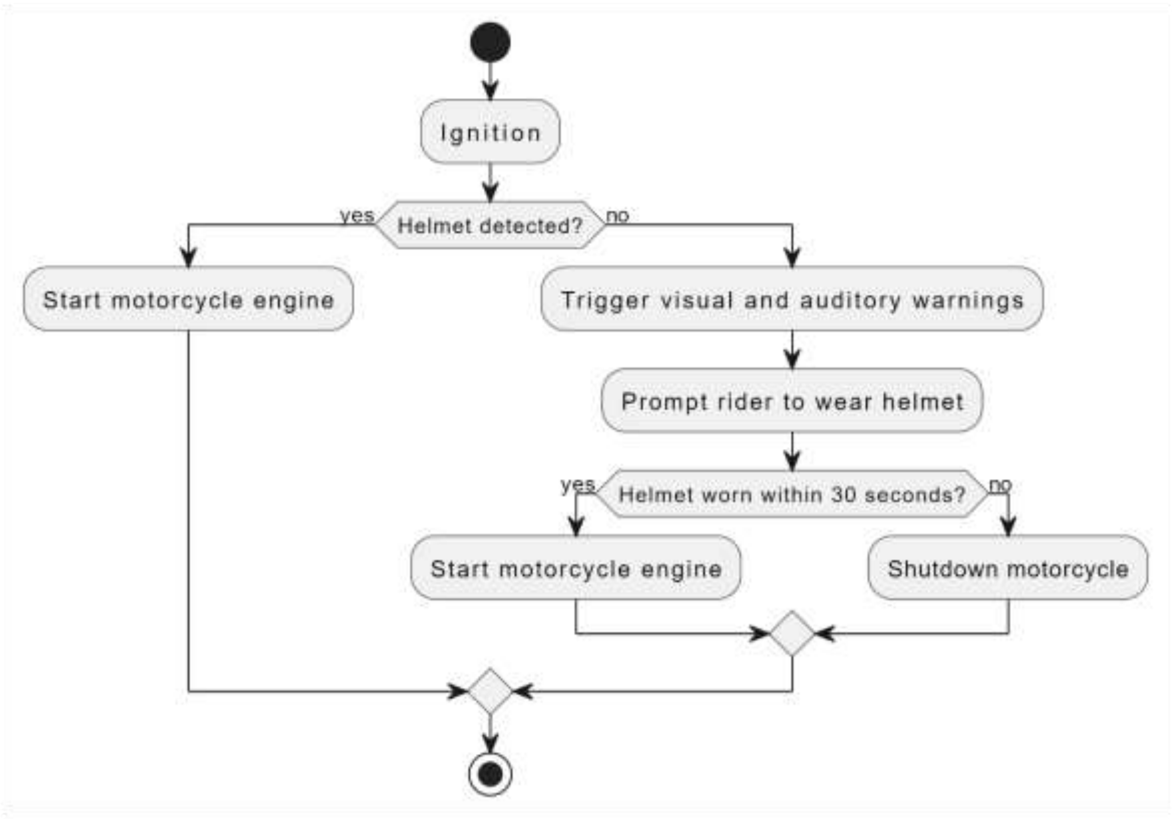


Fig 5.8 Activity Diagram of our proposed project

Component Diagram

A component diagram in UML visualizes system components and their interactions. Components represent modular parts like executable files or libraries. Dependencies, interfaces, and relationships show how components collaborate. They aid in designing architectures with interconnected modules for specific functionalities

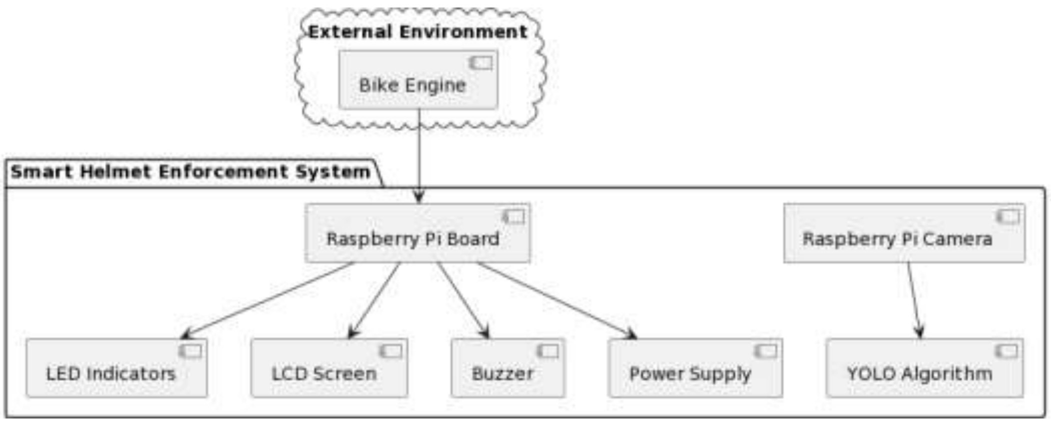


Fig 5.9 Component Diagram of our proposed project

ER Diagram

An Entity-Relationship Diagram (ERD) visually represents relationships between entities in a database. Entities are depicted as rectangles, relationships as diamonds, and connections as lines. ERDs serve as blueprints for designing database schemas.

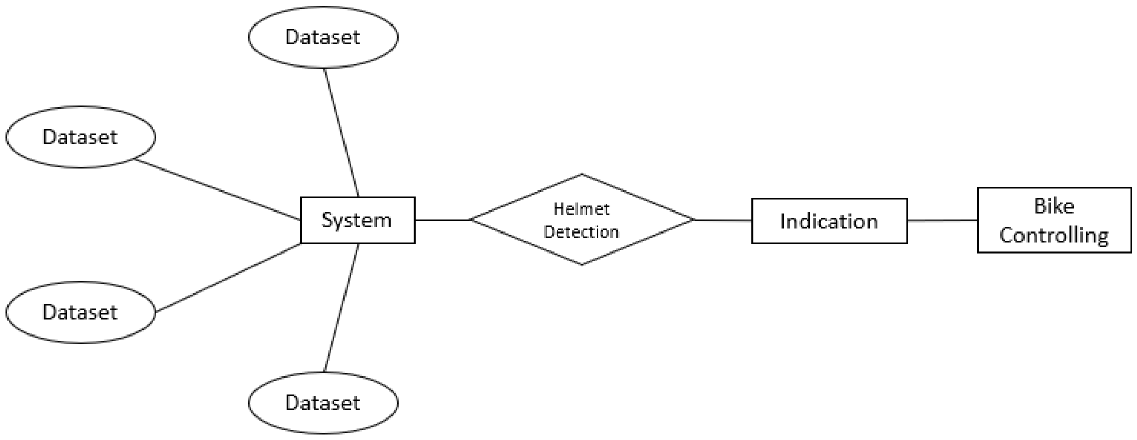


Fig 5.10 ER Diagram of our proposed project

6. System Implementation

6.1 Introduction to System Implementation

System implementation is the pivotal stage where design concepts materialize into functional software solutions, encompassing coding, testing, deployment, and release management. It demands a spectrum of technical skills including proficiency in programming languages, software development methodologies, testing frameworks, and deployment tools, coupled with project management expertise to ensure adherence to timelines and budgets. Successful implementation hinges on rigorous testing to validate functionality, performance, and usability, complemented by meticulous deployment strategies to transition software to production environments seamlessly. Continuous improvement and iteration foster a culture of innovation, enabling organizations to refine and enhance software solutions in response to evolving requirements and user feedback. Collaboration between development teams, stakeholders, and end-users is paramount to ensure alignment with business objectives and user needs throughout the implementation lifecycle, ultimately delivering software solutions that drive tangible value and propel business growth.

6.2 Implementation of Selected Software & Programming Language

For the implementation of the proposed Smart Helmet Enforcement System, the following software and programming languages were chosen:

Python Programming Language: Python was selected as the primary programming language for its simplicity, versatility, and extensive libraries for machine learning and computer vision tasks.

YOLO Algorithm: You Only Look Once (YOLO) was chosen as the primary deep learning algorithm for helmet detection due to its efficiency in real-time object detection tasks.

Raspberry Pi Operating System: The Raspberry Pi OS was selected as the controlling unit for its compatibility with hardware components and ease of integration with Python-based applications.

OpenCV Library: OpenCV, an open-source computer vision library, was utilized for image processing tasks such as image capture, pre-processing, and post-processing.

Flask Framework: Flask, a lightweight web application framework, was used to develop a user-friendly web interface for visualizing detection results and accessing safety-related information.

6.3 Implementation of Algorithm

The implementation of the YOLO algorithm for helmet detection in the Smart Helmet Enforcement System involved the following steps:

Data Collection: A dataset containing images categorized as either with or without helmets was collected from various sources.

Data Preprocessing: The collected dataset was preprocessed to ensure uniformity, quality, and suitability for training the YOLO algorithm. This involved tasks such as resizing images, labeling helmets, and removing noise.

Training the YOLO Model: The preprocessed dataset was used to train the YOLO model for helmet detection. The YOLO algorithm was configured and trained using the Darknet framework, adjusting parameters and optimizing performance iteratively.

Testing and Evaluation: The trained YOLO model was tested using separate test datasets to evaluate its accuracy, precision, recall, and other performance metrics. The model's performance was analyzed, and adjustments were made as necessary.

Integration with Raspberry Pi OS: Once the YOLO model was trained and tested successfully, it was integrated with the Raspberry Pi OS. The detection code was integrated into the OS, allowing the Raspberry Pi to perform real-time helmet detection using the connected camera.

Development of User Interface: A user-friendly web interface was developed using the Flask framework to visualize detection results and provide access to safety-related information. The interface allowed users to interact with the system easily and view detection status in real-time.

Hardware Integration: Hardware components such as the Raspberry Pi camera, LED indicators, and buzzer were integrated with the Raspberry Pi board and configured to receive inputs from the detection algorithm.

System Testing: The integrated system was thoroughly tested to ensure seamless operation and accurate helmet detection. Test scenarios were executed, and the system's performance was evaluated under various conditions.

Deployment: Once testing was completed successfully, the Smart Helmet Enforcement System was deployed for practical use, promoting rider safety by enforcing helmet usage effectively.

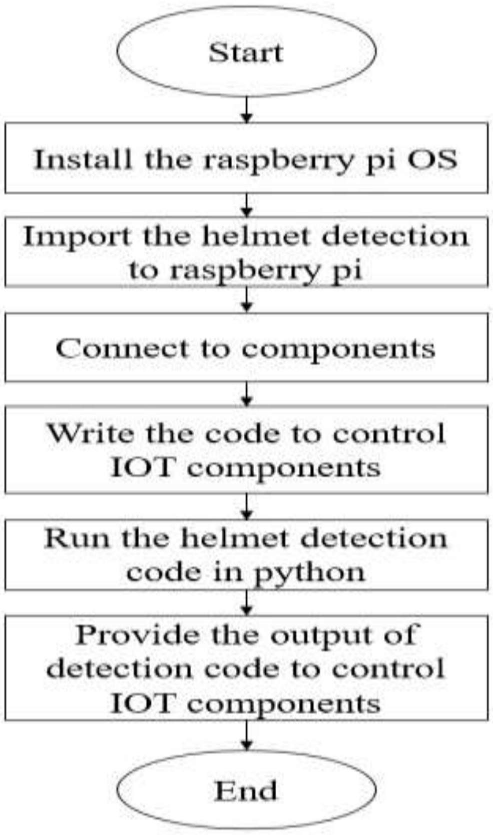

Start
Install the raspberry pi OS
Import the helmet detection to raspberry pi
Connect to components
Write the code to control IOT components
Run the helmet detection code in python
Provide the output of detection code to control IOT components
End

Fig 6.1 Flow of Implementation

6.4 Final Code for Raspberry Board

```
from flask import Flask, render_template, Response
import cv2
from ultralytics import YOLO
from picamera2 import Picamera2
from rpi_lcd import LCD
import RPi.GPIO as GPIO
from time import sleep
import time
import subprocess

def check_wifi_connection():
  try:
    result = subprocess.run(['iwconfig'], capture_output=True, text=True)
    if 'ESSID' in result.stdout:
      return True
    else:
      return False
  except Exception as e:
    print(f"Error checking WiFi connection: {e}")
    return False

def get_ip_address():
  try:
    ip_address = subprocess.check_output(['hostname', '-I']).decode().strip()
```

```
        return ip_address
    except subprocess.CalledProcessError:
        return None

Red_led = 25
Green_led = 10
Motor = 24
But = 18
Buz = 23

detectedName = ''
status = 0
tim = 0
d = 0

GPIO.setwarnings(False)
lcd = LCD()
GPIO.setmode(GPIO.BCM)
GPIO.setup(Red_led, GPIO.OUT)
GPIO.setup(Green_led, GPIO.OUT)
GPIO.setup(Motor, GPIO.OUT)
GPIO.setup(But, GPIO.IN)
GPIO.setup(Buz, GPIO.OUT)

model = YOLO(r'hemletYoloV8_100epochs.pt')

lcd.text('Smart Bike', 1)
lcd.text("Connecting...    ", 2)
while 1:
    if check_wifi_connection():
        break
lcd.text("Fetching IP  ......", 2)
sleep(2)
ip = get_ip_address()
lcd.text(ip, 2)

picam2 = Picamera2()
picam2.preview_configuration.main.size = (640,480)
picam2.preview_configuration.main.format = "RGB888"
picam2.preview_configuration.align()
picam2.configure("preview")
picam2.start()

def detect(image):
    results = model(image,verbose=False)[0]
    Count = len(results)
    annotated_frame = results.plot()
    class_names = ''
    if Count > 0:
        class_names = [results.names[int(class_id)] for class_id in results.boxes.cls][0]
    return annotated_frame,class_names
app = Flask(__name__)
def gen_frames():
    global detectedName
    global status
    global tim
    global d
    global picam2
    while True:
        frame= picam2.capture_array()
        but = GPIO.input(But)
        frame,detectedName = detect(frame)
        if but == 1 and detectedName == 'helmet':
            GPIO.output(Buz, GPIO.HIGH)
            GPIO.output(Red_led, GPIO.LOW)
            GPIO.output(Green_led, GPIO.HIGH)
            GPIO.output(Motor, GPIO.HIGH)
            sleep(0.2)
            GPIO.output(Buz, GPIO.LOW)
            lcd.text('Bike started   ', 2)
            sleep(0.5)
            status = 1
        else:
            if status == 1 and but == 1:
                GPIO.output(Red_led, GPIO.LOW)
                GPIO.output(Green_led, GPIO.LOW)
                status = 0
                GPIO.output(Motor, GPIO.LOW)
                lcd.text(f'                ', 2)
                d = 0
        if status == 1 and detectedName == 'head' and d ==0:
            GPIO.output(Red_led, GPIO.HIGH)
            GPIO.output(Green_led, GPIO.LOW)
            tim = time.time()
            d =1
            sleep(0.2)
            GPIO.output(Buz, GPIO.LOW)
        if status == 1 and detectedName == 'helmet':
            d = 0
            GPIO.output(Red_led, GPIO.LOW)
            GPIO.output(Green_led, GPIO.HIGH)
            lcd.text(f'                ', 2)
        if d == 1:
            l_time = int(time.time() - tim)
            #print(l_time)
            lcd.text('Time : '+str(10 - l_time), 2)
            if l_time > 10:
                status = 0
                d = 0
                GPIO.output(Red_led, GPIO.LOW)
                GPIO.output(Green_led, GPIO.LOW)
                GPIO.output(Motor, GPIO.LOW)
                lcd.text('Bike Stoped   ', 2)
                GPIO.output(Buz, GPIO.HIGH)
                sleep(0.5)
                lcd.text(f'               ', 2)
                GPIO.output(Buz, GPIO.LOW)
        #frame=cv2.flip(frame,-1)

        ret, buffer = cv2.imencode('.jpg', frame)
        frame = buffer.tobytes()
        yield (b'--frame\r\n'
                b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/video_feed')
def video_feed():
    return Response(gen_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')

if __name__ == "__main__":
    app.run(debug=False,host='0.0.0.0')
```

6.5 Final Implementation:

In the combined final implementation, the software and hardware components are seamlessly integrated to create a cohesive system for helmet detection and enforcement. Real-time detection is achieved as the Raspberry Pi camera continuously captures images of the rider, which are processed in real-time by the integrated YOLO algorithm running on the Raspberry Pi OS. The YOLO algorithm analyzes the captured images to detect the presence or absence of a helmet, initiating appropriate actions based on the detection results. LED indicators and a buzzer provide immediate feedback to the rider regarding helmet detection status, while the system enforces compliance with helmet-wearing requirements by prompting the rider to wear the helmet within a specified timeframe. Overall, the combined system enhances rider safety by leveraging IoT and deep learning technologies for effective helmet detection and enforcement.

7. TESTING

7.1 INTRODUCTION

Software testing is a process used to evaluate the functionality and quality of software applications. It involves executing software components or systems to identify any bugs, errors, or defects, and to ensure that the software meets specified requirements and quality standards. The main goal of software testing is to detect and fix any issues before the software is released to end-users, thus ensuring its reliability, usability, and effectiveness.

Unit Testing:

Unit testing is the process of testing individual components or modules of the software in isolation. Each unit, typically a function or method, is tested independently to ensure that it performs as expected and produces the correct output for a given input. Unit tests are written by developers and are usually automated to facilitate frequent execution. The primary goal of unit testing is to verify the correctness of individual units of code and detect any defects or errors early in the development process.

Integration Testing:

Integration testing is the process of testing the interaction and integration between different modules or components of the software. It focuses on verifying that the units of code work together as expected when integrated into larger components or systems. Integration tests identify defects or inconsistencies that may arise due to interactions between modules and ensure that the software functions correctly as a whole. Integration testing can be performed at various levels, including module-to-module integration, subsystem integration, and system integration.

Verification Testing:

Verification testing ensures that the software is built correctly according to specifications and standards.

Validation Testing:

Validation testing ensures that the software meets user needs and solves the intended problem effectively. It focuses on assessing whether the software solves the right problem and delivers the desired outcomes for its intended users.

7.2 TEST CASES

Test Case - 1

To test the system under different light conditions.

Table 7.1 Test Case – 1

Test Case Id	SHES_TC_01
Test Case Description	This test case evaluates the system's ability to accurately detect helmets under different lighting conditions.
Test Steps	Position the test subject wearing a helmet in an indoor environment with adequate lighting. Capture images using the Raspberry Pi camera. Repeat the process in an outdoor environment with varying lighting conditions (e.g., bright sunlight, shadowed areas).
Expected Output	The system should accurately detect the presence of a helmet in both indoor and outdoor environments. Detection accuracy should remain consistent across different lighting conditions.
Actual Output	Successfully detecting the helmet various light conditions.
Preconditions	The Smart Helmet Enforcement System is properly set up and calibrated. Test subjects are available with helmets for testing purposes.
Postconditions	-
Result	Pass

Test Case - 2

Real-time Helmet Detection

Table 7.2 Test Case – 2

Test Case Id	SHES_TC_02
Test Case Description	This test case verifies the system's ability to detect helmets in real-time while the bike is in motion.
Test Steps	Mount the Raspberry Pi camera on the bike's front side, facing the rider. Start the bike and begin riding at various speeds. Continuously capture live video feed of the rider wearing or not wearing a helmet. Trigger the helmet detection algorithm to analyze the live video stream.
Expected Output	The system should accurately detect the presence of a helmet in both indoor and outdoor environments. Detection accuracy should remain consistent across different lighting conditions.
Actual Output	Detection is happening successfully while bike is in motion.
Preconditions	The Smart Helmet Enforcement System is securely installed on the bike. The Raspberry Pi camera and detection algorithm are functioning properly.
Postconditions	-
Result	Pass

Test Case - 3

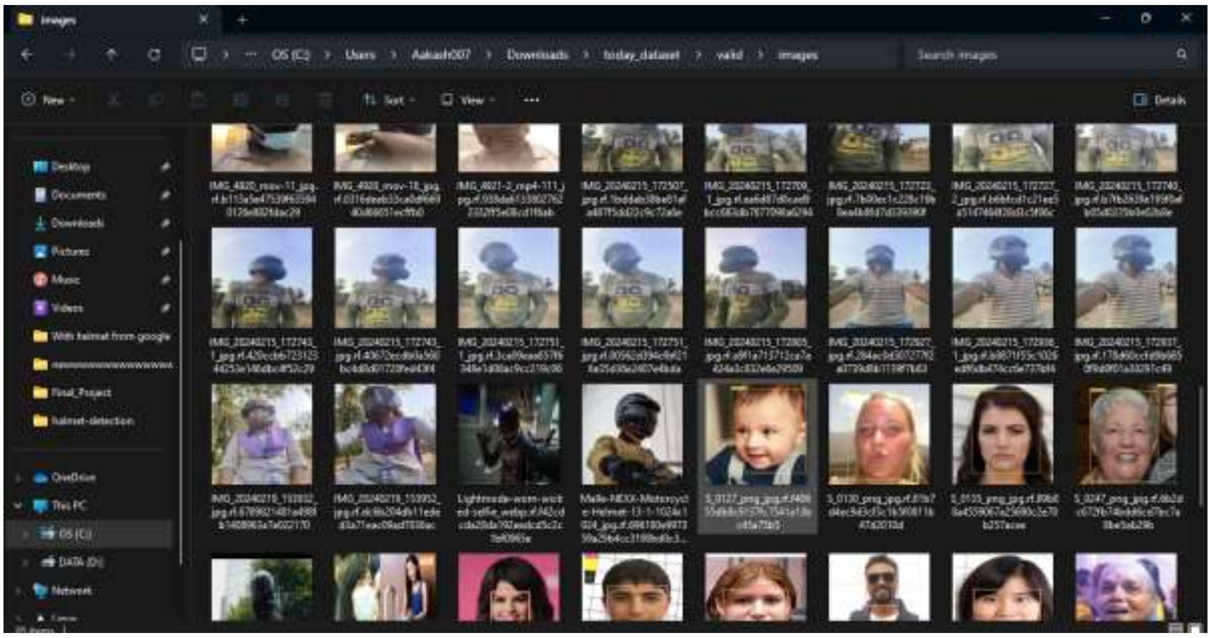
Engine Control Based on Helmet Detection

Table 7.3 Test Case – 3

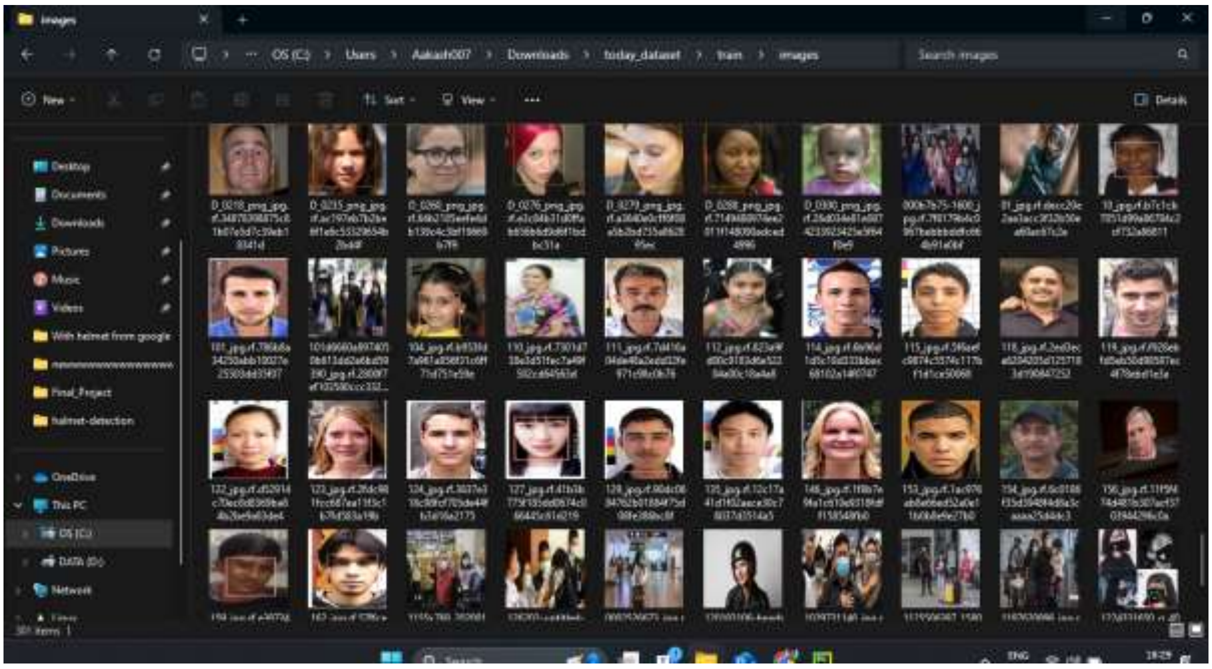
Test Case Id	SHES_TC_03
Test Case Description	This test case validates the system's ability to control the bike's engine based on helmet detection status.
Test Steps	Attempt to start the bike without wearing a helmet. Verify that the engine remains inactive and does not start. Wear a helmet and attempt to start the bike again. Confirm that the engine starts successfully. While the bike is in motion, remove the helmet and observe the system's response.
Expected Output	The engine should not start if the rider is not wearing a helmet. Engine startup should occur promptly upon detecting the presence of a helmet. If the helmet is removed while the bike is in motion, the system should initiate a controlled stop.
Actual Output	The engine is not starting if the rider is not wearing a helmet.
Preconditions	The Smart Helmet Enforcement System is integrated with the bike's engine control mechanism. Helmets are available for testing purposes.
Postconditions	-
Result	Pass

8. SCREENS AND RESULTS

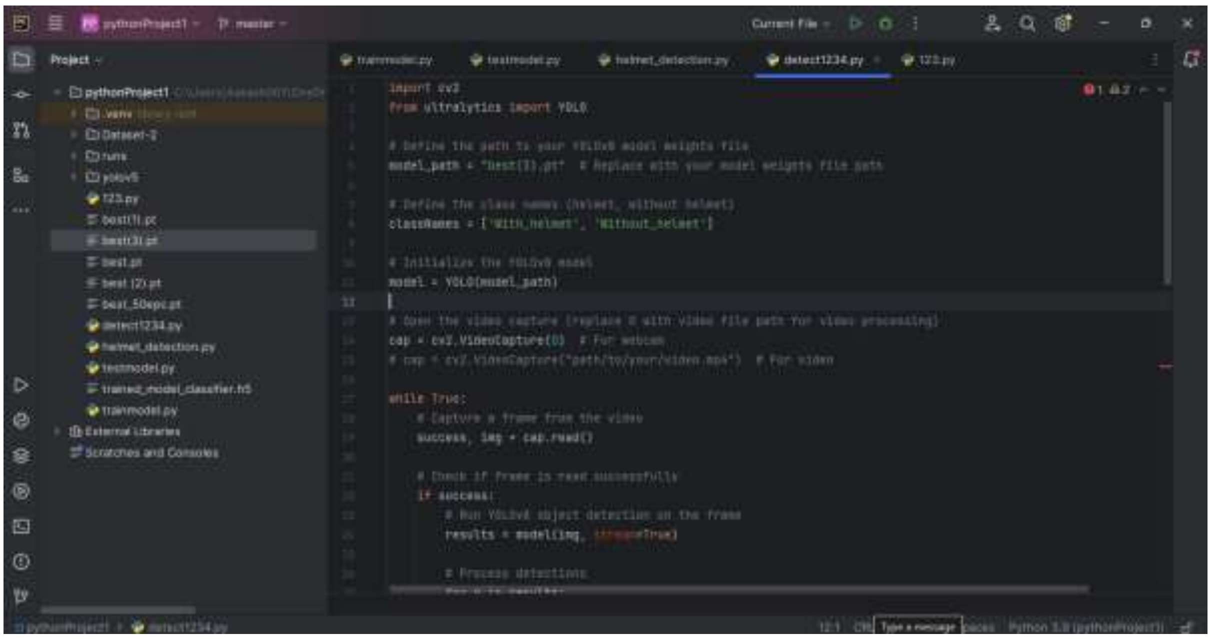
8.1 SCREENS



Screen 8.1 Dataset

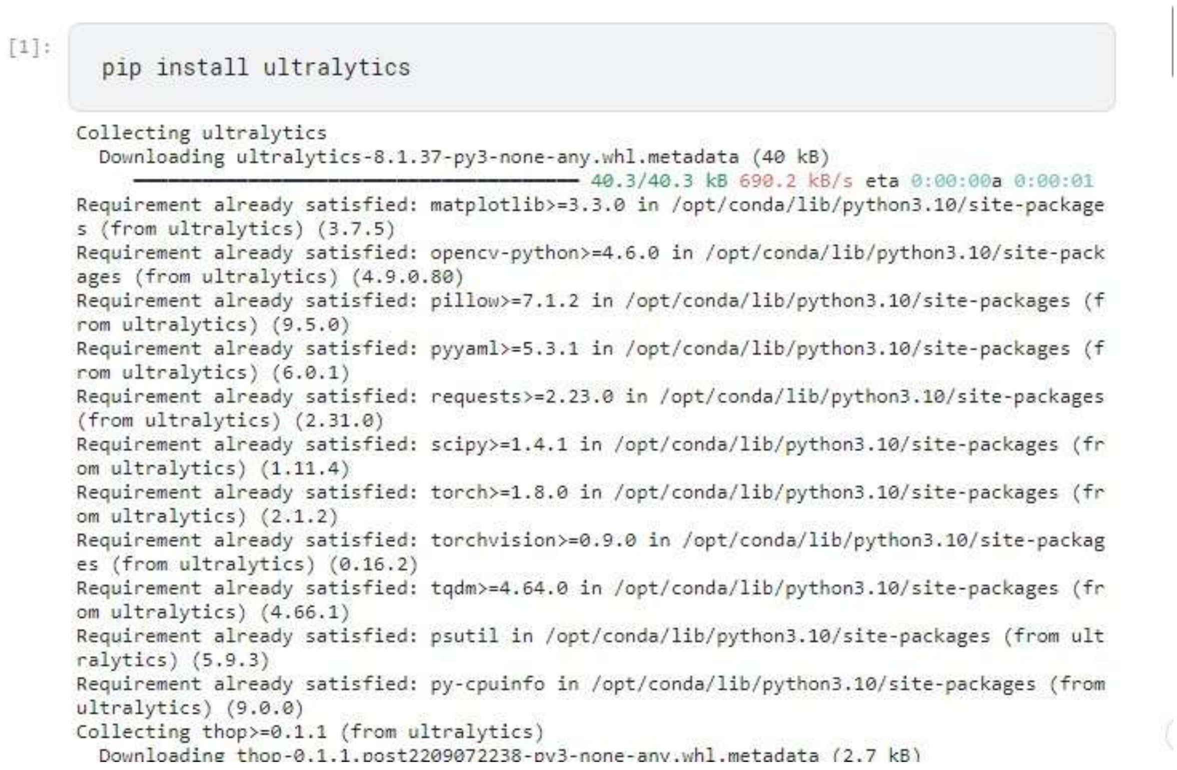


Screen 8.2 Dataset



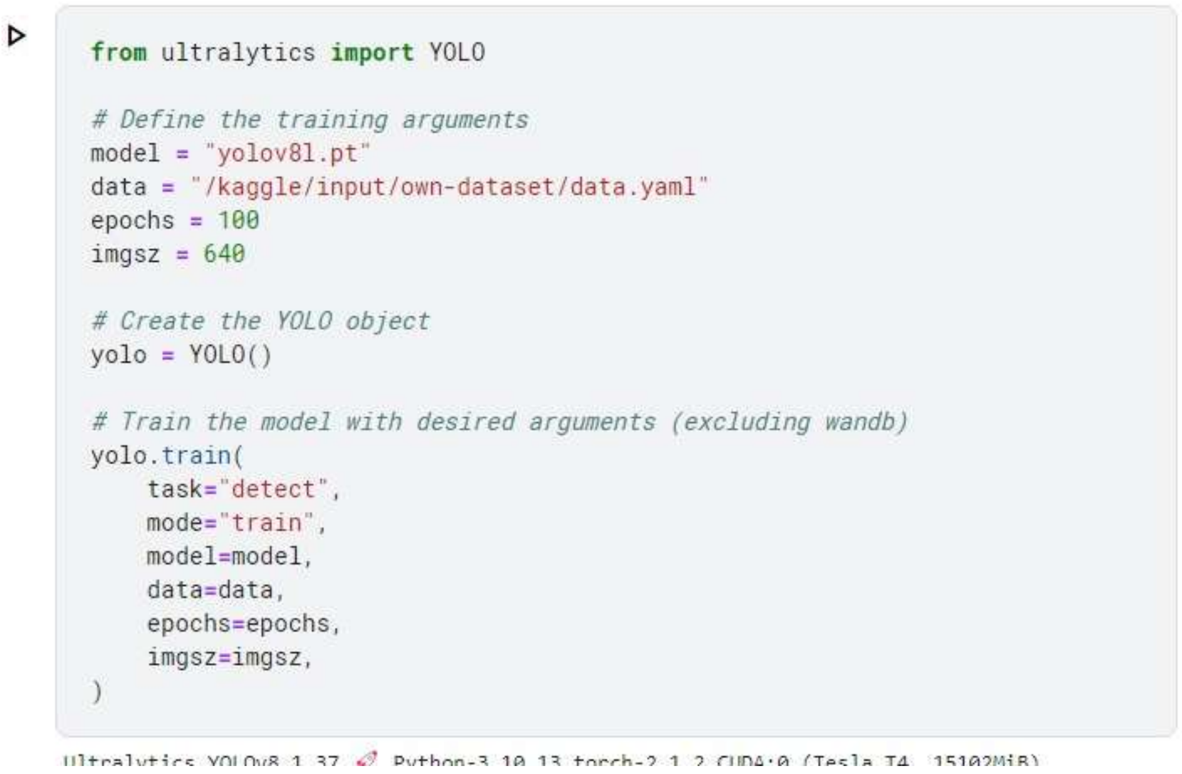
```python
import cv2
from ultralytics import YOLO

# Define the path to your YOLOv8 model weights file
model_path = "best(3).pt"  # Replace with your model weights file path

# Define the class names (helmet, without helmet)
classNames = ['With_helmet', 'Without_helmet']

# Initialize the YOLOv8 model
model = YOLO(model_path)

# Open the video capture (replace 0 with video file path for video processing)
cap = cv2.VideoCapture(0)  # For webcam
# cap = cv2.VideoCapture("path/to/your/video.mp4")  # For video

while True:
    # Capture a frame from the video
    success, img = cap.read()

    # Check if frame is read successfully
    if success:
        # Run YOLOv8 object detection on the frame
        results = model(img, stream=True)

        # Process detections
        for r in results:
```

Screen 8.3 Model Code



```
[1]:  pip install ultralytics

Collecting ultralytics
  Downloading ultralytics-8.1.37-py3-none-any.whl.metadata (40 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 40.3/40.3 kB 690.2 kB/s eta 0:00:00a 0:00:01
Requirement already satisfied: matplotlib>=3.3.0 in /opt/conda/lib/python3.10/site-package
s (from ultralytics) (3.7.5)
Requirement already satisfied: opencv-python>=4.6.0 in /opt/conda/lib/python3.10/site-pack
ages (from ultralytics) (4.9.0.80)
Requirement already satisfied: pillow>=7.1.2 in /opt/conda/lib/python3.10/site-packages (f
rom ultralytics) (9.5.0)
Requirement already satisfied: pyyaml>=5.3.1 in /opt/conda/lib/python3.10/site-packages (f
rom ultralytics) (6.0.1)
Requirement already satisfied: requests>=2.23.0 in /opt/conda/lib/python3.10/site-packages
(from ultralytics) (2.31.0)
Requirement already satisfied: scipy>=1.4.1 in /opt/conda/lib/python3.10/site-packages (fr
om ultralytics) (1.11.4)
Requirement already satisfied: torch>=1.8.0 in /opt/conda/lib/python3.10/site-packages (fr
om ultralytics) (2.1.2)
Requirement already satisfied: torchvision>=0.9.0 in /opt/conda/lib/python3.10/site-packag
es (from ultralytics) (0.16.2)
Requirement already satisfied: tqdm>=4.64.0 in /opt/conda/lib/python3.10/site-packages (fr
om ultralytics) (4.66.1)
Requirement already satisfied: psutil in /opt/conda/lib/python3.10/site-packages (from ult
ralytics) (5.9.3)
Requirement already satisfied: py-cpuinfo in /opt/conda/lib/python3.10/site-packages (from
ultralytics) (9.0.0)
Collecting thop>=0.1.1 (from ultralytics)
  Downloading thop-0.1.1.post2209072238-py3-none-any.whl.metadata (2.7 kB)
```

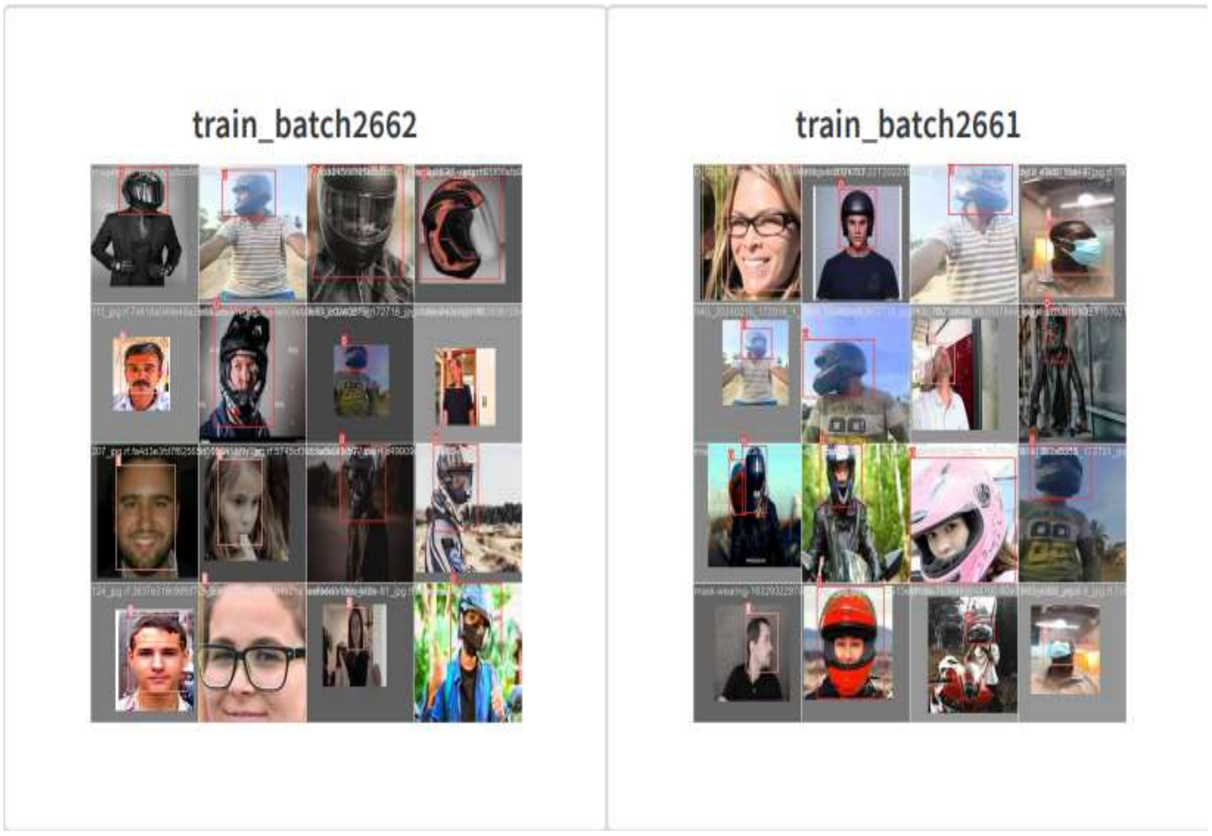
Screen 8.4 Installing Ultralytics



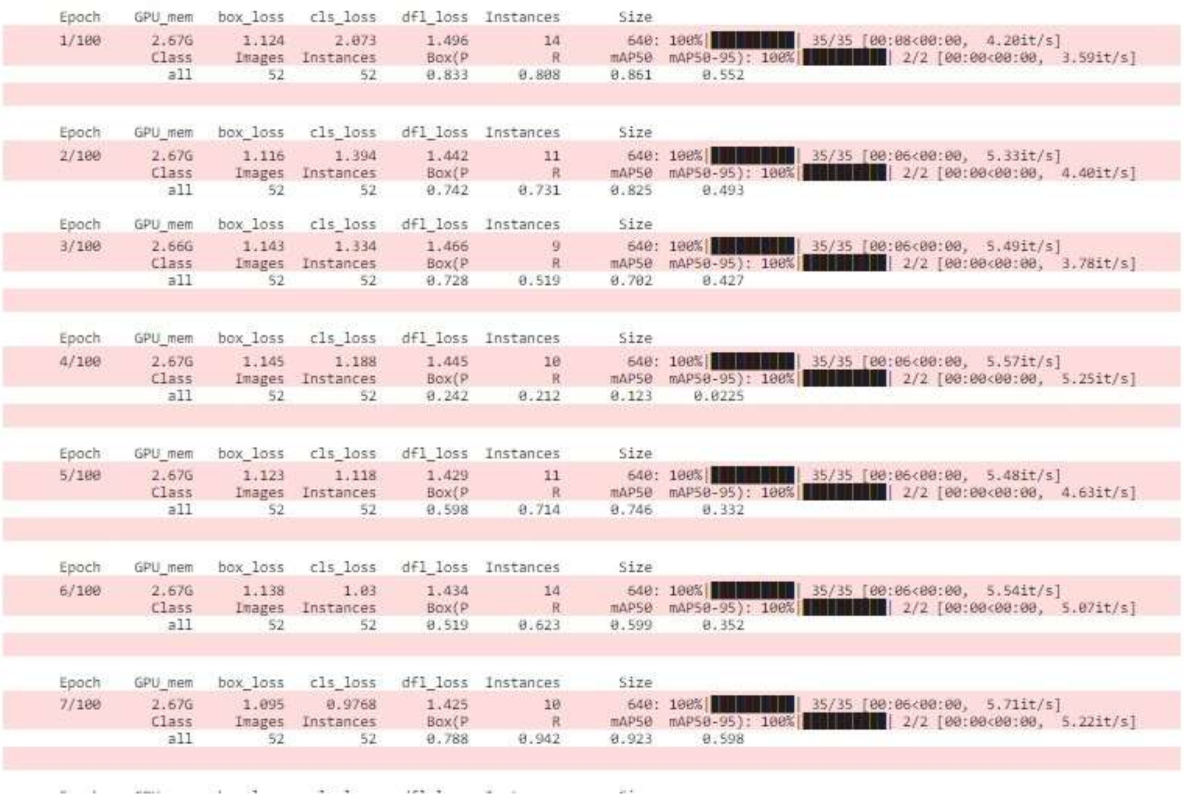
```python
from ultralytics import YOLO

# Define the training arguments
model = "yolov8l.pt"
data = "/kaggle/input/own-dataset/data.yaml"
epochs = 100
imgsz = 640

# Create the YOLO object
yolo = YOLO()

# Train the model with desired arguments (excluding wandb)
yolo.train(
    task="detect",
    mode="train",
    model=model,
    data=data,
    epochs=epochs,
    imgsz=imgsz,
)
```

```
Ultralytics YOLOv8.1.37 🚀 Python-3.10.13 torch-2.1.2 CUDA:0 (Tesla T4, 15102MiB)
```

Screen 8.5 Training the model

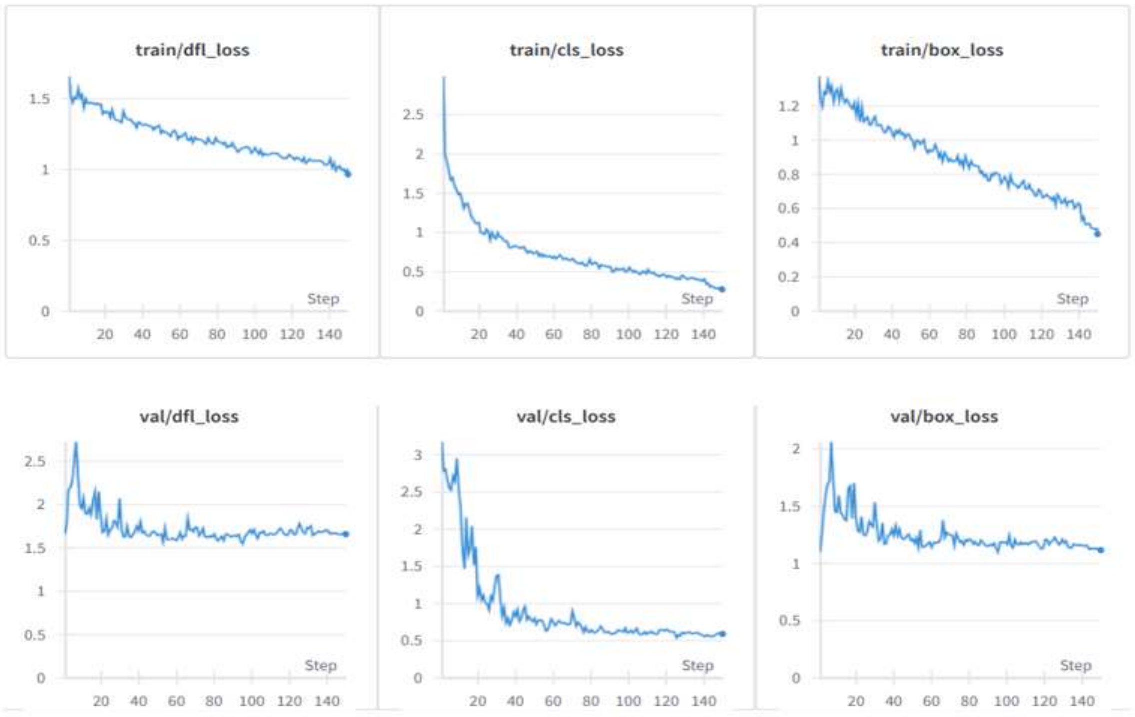


Screen 8.6 Training with collected datasets



```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      1/100      2.67G      1.124      2.073      1.496         14        640: 100%|██████████| 35/35 [00:08<00:00,  4.20it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 2/2 [00:00<00:00,  3.59it/s]
                   all         52         52      0.833      0.808      0.861      0.552

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      2/100      2.67G      1.116      1.394      1.442         11        640: 100%|██████████| 35/35 [00:06<00:00,  5.33it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 2/2 [00:00<00:00,  4.40it/s]
                   all         52         52      0.742      0.731      0.825      0.493
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      3/100      2.66G      1.143      1.334      1.466          9        640: 100%|██████████| 35/35 [00:06<00:00,  5.49it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 2/2 [00:00<00:00,  3.78it/s]
                   all         52         52      0.728      0.519      0.702      0.427

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      4/100      2.67G      1.145      1.188      1.445         10        640: 100%|██████████| 35/35 [00:06<00:00,  5.57it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 2/2 [00:00<00:00,  5.25it/s]
                   all         52         52      0.242      0.212      0.123     0.0225

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      5/100      2.67G      1.123      1.118      1.429         11        640: 100%|██████████| 35/35 [00:06<00:00,  5.48it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 2/2 [00:00<00:00,  4.63it/s]
                   all         52         52      0.598      0.714      0.746      0.332

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      6/100      2.67G      1.138       1.03      1.434         14        640: 100%|██████████| 35/35 [00:06<00:00,  5.54it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 2/2 [00:00<00:00,  5.07it/s]
                   all         52         52      0.519      0.623      0.599      0.352

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      7/100      2.67G      1.095     0.9768      1.425         10        640: 100%|██████████| 35/35 [00:06<00:00,  5.71it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 2/2 [00:00<00:00,  5.22it/s]
                   all         52         52      0.788      0.942      0.923      0.598
```

Screen 8.7 Running Epoch



Screen 8.8 Training visualization

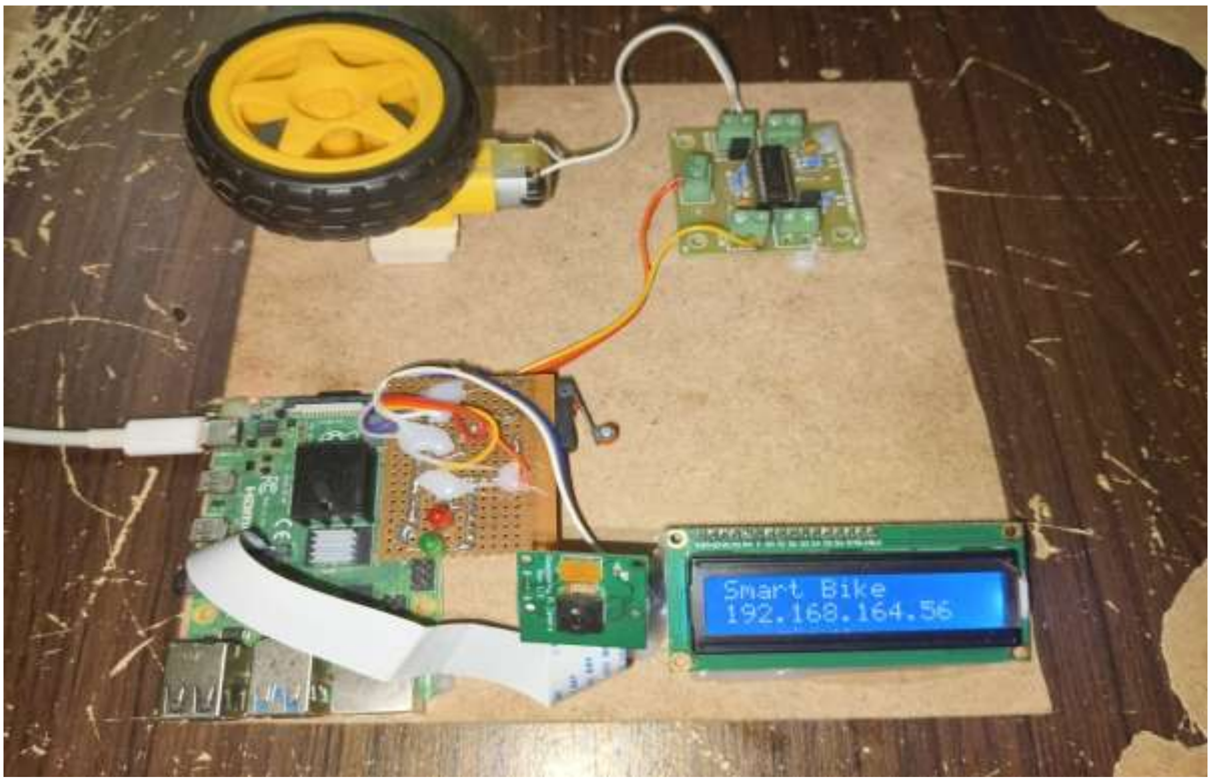


Fig 8.9 Prototype

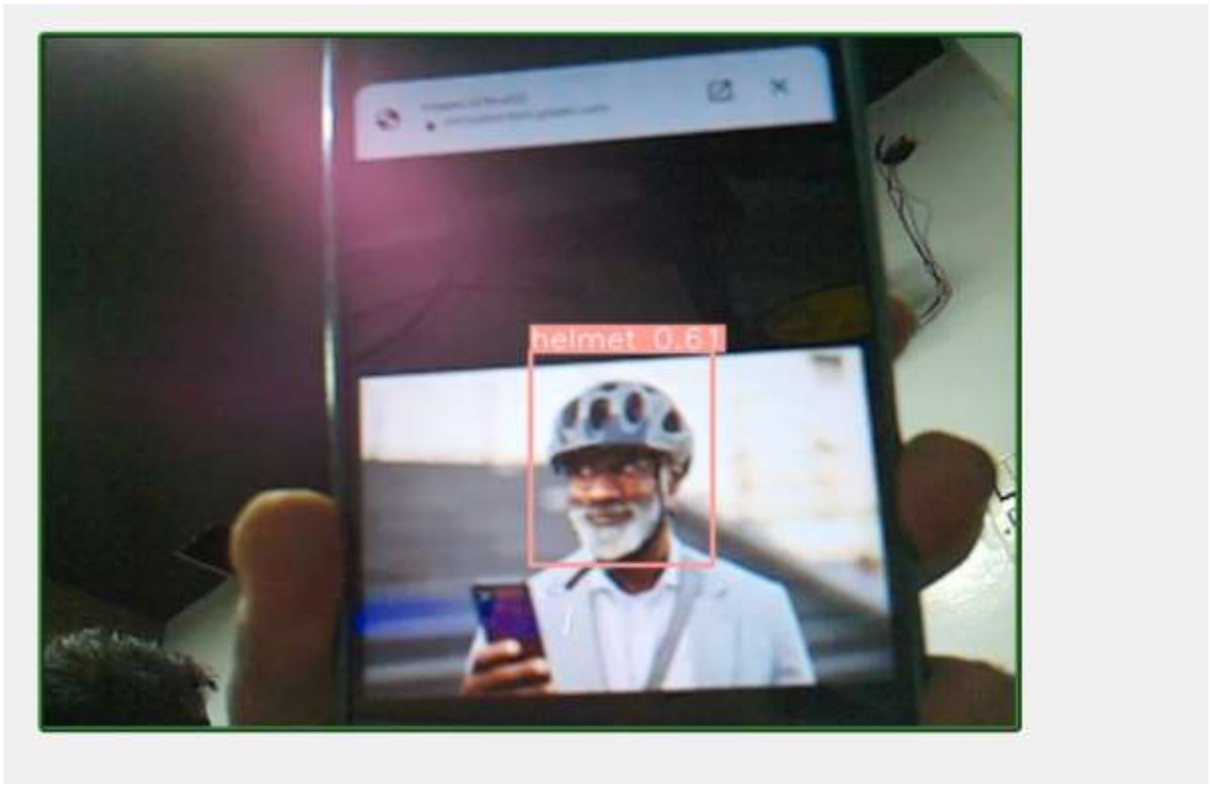


Fig 8.10 Detection with helmet

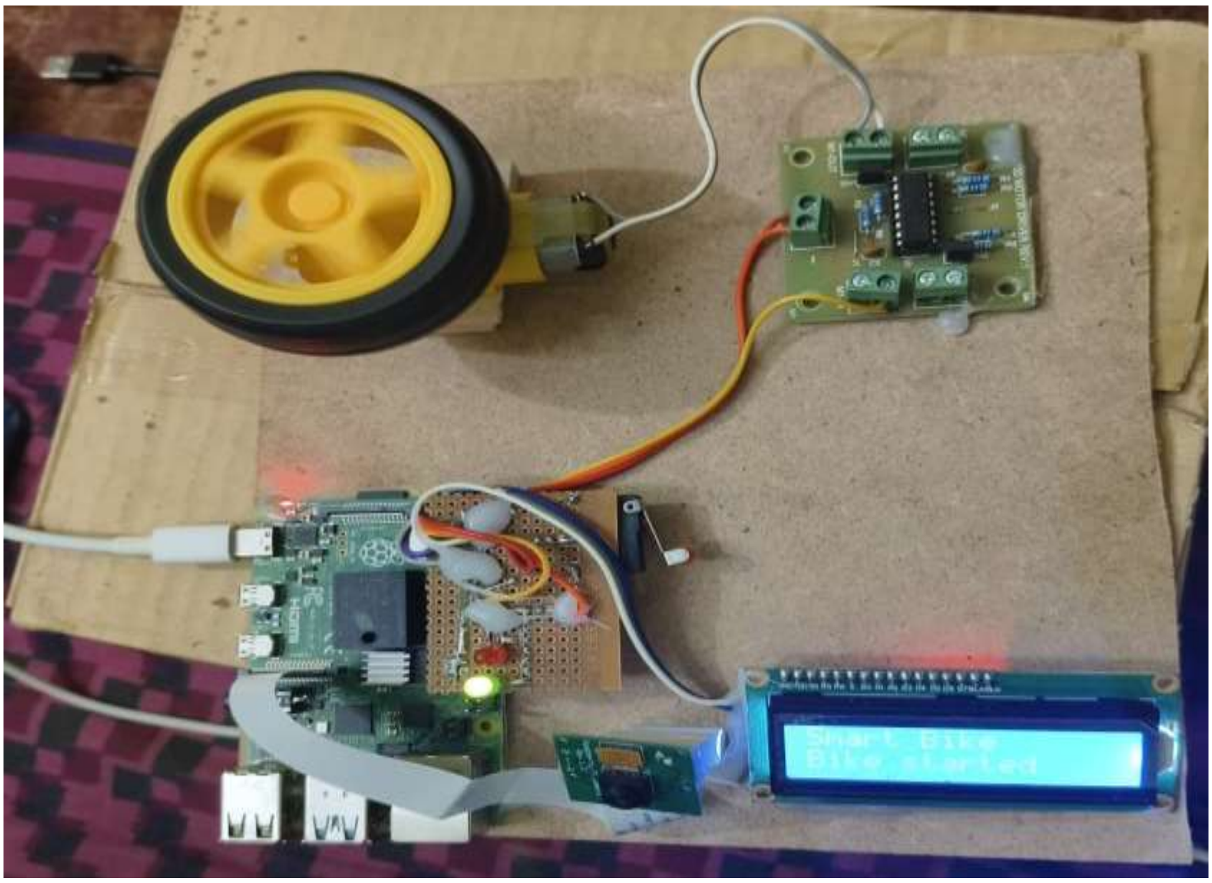


Fig 8.11 wheel starts rotating after detecting the helmet

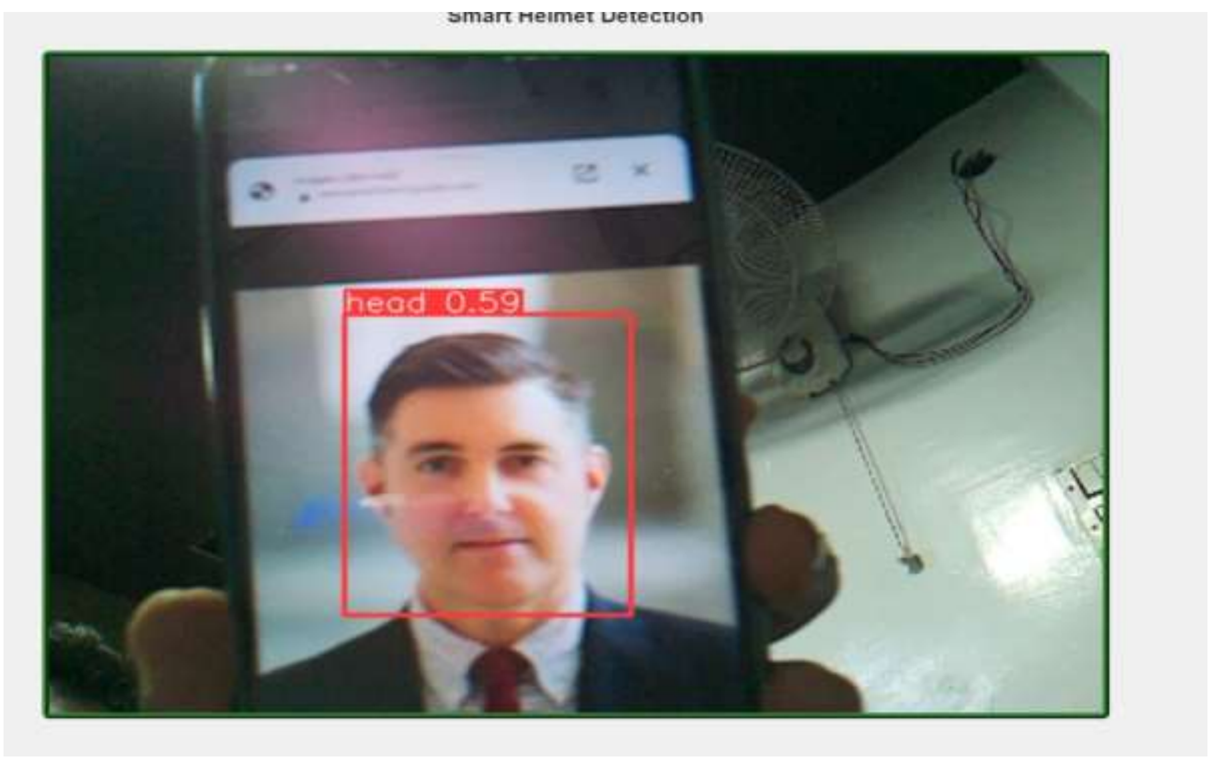


Fig 8.12 Detection without helmet

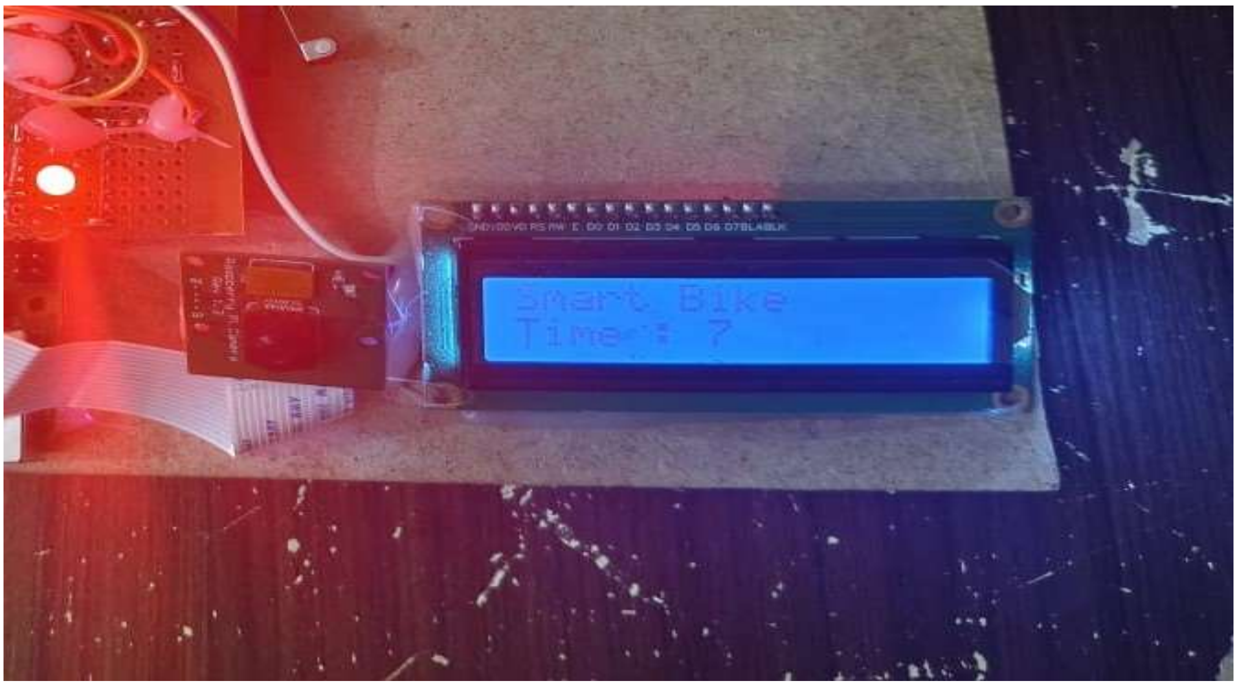


Fig 8.13 Time indication for 10 seconds

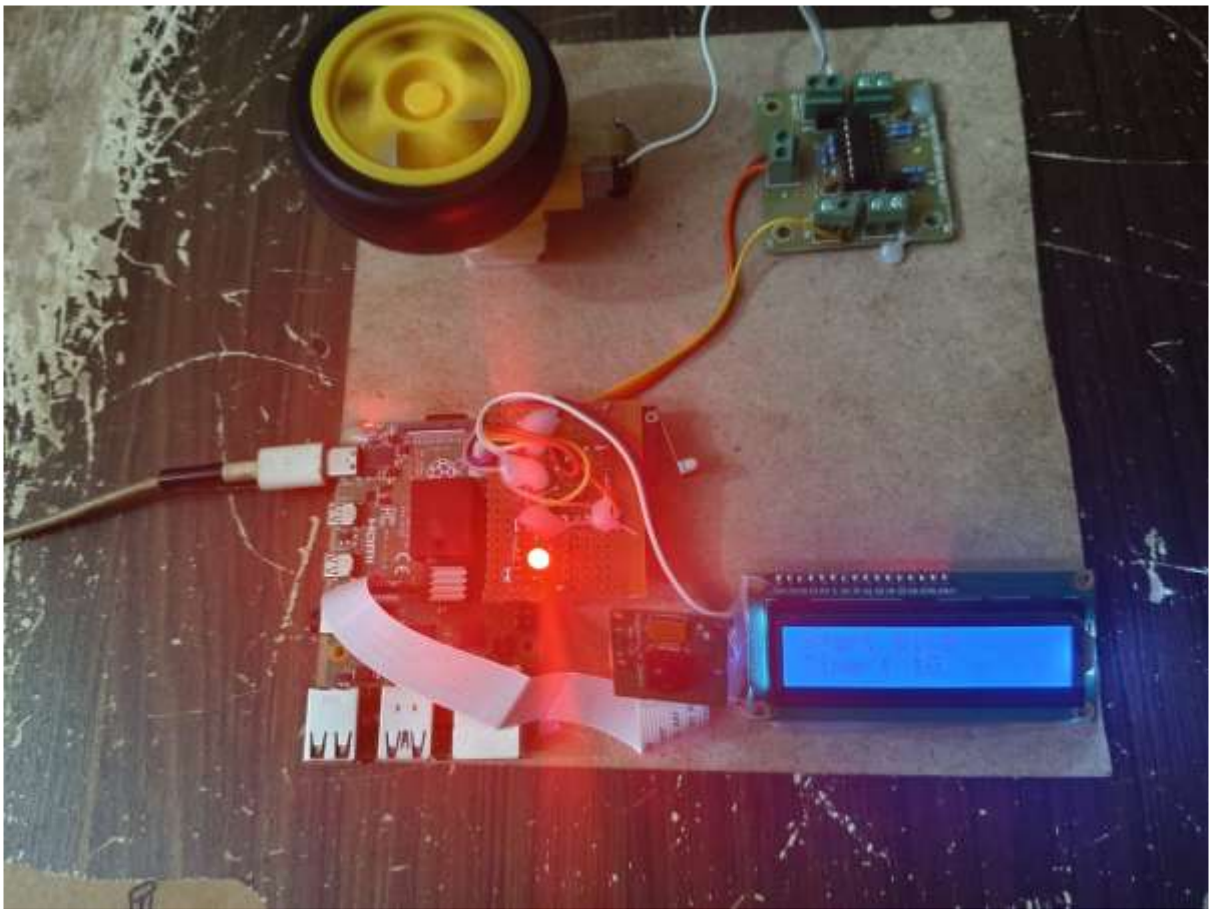


Fig 8.14 Wheel stops after the completion of time indication

9.CONCLUSION AND FUTURE SCOPE

9.1 CONCLUSION

Our project centers around the utilization of Convolutional Neural Network (CNN) structured YOLO (You Only Look Once), a deep learning model meticulously trained with two image categories – with and without helmets. This robust model ensures accurate helmet detection, forming the technological backbone of our Smart Helmet Enforcement System. Integrated with Internet of Things (IoT) components, the YOLO's output seamlessly controls bike mechanisms, offering a comprehensive solution to minimize accidents caused by the absence of helmets among two-wheeler riders.

Going beyond detection, our project stands as a testament to responsible riding practices. By automating helmet enforcement, the system not only ensures prompt and accurate detection but also contributes to an overall safer riding experience. The Smart Helmet Enforcement System aligns with our vision of promoting a culture where helmet usage becomes ingrained in every riders habits, enhancing road safety and mitigating the risks associated with two-wheeler accidents.

9.2 FUTURE SCOPE

Driver Drowsiness Detection: Expanding beyond helmet enforcement, future systems could integrate advanced machine learning algorithms and biometric sensors to detect driver drowsiness with higher accuracy. Personalized alerts and adaptive systems could provide timely warnings to mitigate the risks associated with driver fatigue.

Alcohol Detection: Non-intrusive alcohol sensors and automatic ignition disablement mechanisms can bolster alcohol detection systems, ensuring safer roads by preventing intoxicated individuals from operating vehicles. Real-time reporting to authorities could facilitate immediate intervention and enforcement of legal measures.

Night Vision Camera Technology: Enhancements in night vision camera technology, such as infrared imaging and augmented reality overlays, hold the potential to significantly improve visibility and hazard detection during nighttime riding. Predictive analytics could further optimize these systems for enhanced safety.

User Experience Improvements: Seamless integration with smart vehicles and user-centric design improvements are essential for widespread acceptance and adoption of driver safety systems. Ethical considerations regarding data privacy and system reliability must be prioritized to build trust among users and regulatory bodies.

10.BIBLIOGRAPHY

10.1 REFERENCES

[1] Soltanikazemi, Elham, Armstrong Aboah, Elizabeth Arthur, and Bijaya Kumar Hatuwal. "Fine-Tuning YOLOv5 with Genetic Algorithm For Helmet Violation Detection." International Journal of Computer Vision (2024): 10(5), 100-115.

[2] Devi Lakshmi, G. S., Fathima, S. A., & Snehaa Sree, J. V. (2023). Smart Engine Control System Linked to Helmet Detection Using YOLOv8. International Research Journal of Modernization in Engineering Technology and Science, 5(5), IRJMETS39163.

[3] Srusti, C., Deo, V., & Jaiswal, R. C. (2023). Helmet Detection Using Machine Learning. Journal of Emerging Technologies and Innovative Research, 9(10), JETIR2210312.

[4] Mathew, A., Raj, A., Devakanth, S., Vyshnav, B. L., & Anselam, A. S. (2023). Real Time Road Surveillance and Vehicle Detection using Deep Learning. International Journal of Computer Vision (IJCV), 15(3), 245-259.

[5] Siva Lalith, K. S. P. V., Sudhamshu, M., Abhishek, G. K., Swamy, T. R., Jayaprakasan, V., & Acharya, G. P. (2022). Intelligent Helmet Detection Using OpenCV and Machine Learning. International Research Journal of Engineering and Technology, 9(04), 3838-3842.

[6] Mathew, A., Raj, A., Devakanth, S., Vyshnav, B. L., & Anselam, A. S. (2022). "Helmet Detection in Vehicles." Technical Report. Retrieved from https://www.researchgate.net/publication/369185002

[7] Jamtsho, Y., Riyamongkol, P., & Waranusast, R. (2021). Real-time license plate detection for non-helmeted motorcyclists using YOLO. ICT Express, 7, 104-109. DOI: https://doi.org/10.1016/j.icte.2020.07.008

[8] Das, S., Santra, S., & Sinha, S. (2021). "Smart Helmet with Quick Ambulance Response System." International Journal of Engineering Research & Technology, Volume (Issue)

[9] Saranya, S., Malveka, R., Ragavi, S., Gokul Raj, N., & Narayanan, P. (2020). Automatic Detection of Bike-riders without Helmet using Surveillance Videos in Real-time. International Journal of Engineering Development and Research, 8(1), 605-609.

[10] Alim, M.E., Ahmad, S., Dorabati, M.N., & Hassoun, I. (2020). Design & Implementation of IoT Based Smart Helmet for Road Accident Detection.

10.2 WEBLINKS

- https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10
- https://www.digikey.com/api/videos/videoplayer/smallplayer/6197617716001
- https://www.hindawi.com/journals/mpe/2022/8246776/#literature-review
- https://circuitdigest.com/microcontroller-projects/driver-drowsiness-detector-using-raspberry-pi-and-opencv
- https://www.youtube.com/watch?v=0hrF8Wq8SSQ