A

MAJOR PROJECT REPORT ON

AI BASED SURVEILLANCE SYSTEM FOR DETECTION OF VEHICLES WITHOUT HELMET USING YOLO TECHNOLOGY

Submitted in partial fulfilment of the requirement for the award of degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

SUBMITTED BY

VELISHALA GANESH **218R1A04C4**

Under the Esteemed Guidance of

Dr. K. Sravan Abhilash

Associate Professor

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Approved by AICTE, Affiliated to JNTU Hyderabad, Accredited by NBA& NAAC)
Kandlakoya (V), Medchal (M), Telangana –501401

2024-2025

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Approved by AICTE, Affiliated to JNTU Hyderabad, Accredited by NBA& NAAC)
Kandlakoya (V), Medchal (M), Telangana –501401

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify a major -project work entitled **"AI BASED SURVEILLANCE SYSTEM FOR DETECTION OF VEHICLES WITHOUT HELMET USING YOLO TECHNOLOGY"** is being submitted by **V. GANESH** bearing Roll No:**218R1A04C4** in BTech IV-II semester, Electronics and Communication Engineering is a record Bonafide work carried out by then during the academic year 2024-25.

INTERNAL GUIDE
Dr. K. Sravan Abhilash

HEAD OF THE DEPARTMENT
Dr. Suman Mishra

EXTERNAL EXAMINER

ACKNOWLEDGEMENTS

I sincerely thank the management of our college **CMR Engineering College** for providing required facilities during our project work. We derive great pleasure in expressing our sincere gratitude to our Principal **Dr. A. S. Reddy** for his timely suggestions, which helped us to complete the project work successfully. It is the very auspicious moment we would like to express our gratitude to **Dr**. **SUMAN MISHRA,** Head of the Department, ECE for his consistent encouragement during the progress of this project.

I take it as a privilege to thank our major project coordinator **Dr.T. SATYANARAYANA**, Professor, Department of ECE for the ideas that led to complete the project work and we also thank him for his continuous guidance, support and unfailing patience, throughout the course of this work. We sincerely thank our project internal guide **Dr. K. Sravan Abhilash,** Associate Professor of ECE for guidance and encouragement in carrying out this project work.

DECLARATION

I hereby declare that a major-project entitled **"AI BASED SURVEILLANCE SYSTEM FOR DETECTION OF VEHICLES WITHOUT HELMET USING YOLO TECHNOLOGY"** is the work done by us in campus at **CMR ENGINEERING COLLEGE,** Kandlakoya during the academic year 2024-2025 and is submitted as major project in partial fulfilment of the requirements for the award of degree of **BACHELOR OF TECHNOLOGY** in **ELECTRONICS AND COMMUNICATION ENGINEERING FROM JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD.**

V. GANESH **218R1A04C4**

ABSTRACT

The project focuses on developing a machine learning-based surveillance system designed to enhance traffic safety by detecting bike riders without helmets and identifying triple riders in real-time. Leveraging advanced computer vision techniques, the system utilizes Convolutional Neural Networks (CNNs) and state-of-the-art detection algorithms such as YOLO (You Only Look Once) and Faster R-CNN to analyse live video feeds from traffic cameras.

With high accuracy (96.66%) and a low false alarm rate of 0.5%, the system reliably identifies violations, even in complex traffic scenarios. It processes video feeds frame by frame, detecting helmet-less riders and triple riding through automated classification and object detection. Integration with pre-trained models like ResNet and EfficientNet further enhances its performance.

The adaptive learning mechanism enables continuous improvement through training on new datasets, ensuring the system remains robust across diverse environments. Designed for seamless integration into existing traffic management systems, this solution offers real-time monitoring, automated detection, and scalable deployment, significantly contributing to safer roadways and improved traffic regulation.

In addition to detecting helmet violations and triple riding, the system can be further enhanced by integrating license plate recognition to automate fine generation. By leveraging Optical Character Recognition (OCR) techniques, the system can extract vehicle registration numbers from detected violators, streamlining law enforcement and reducing the need for manual intervention. This feature enables authorities to issue automated penalty notifications, ensuring a more efficient and transparent traffic monitoring process. Moreover, the system's scalability allows it to be deployed across multiple urban locations with varying traffic densities, adapting to different lighting conditions and environmental factors for consistent performance.

CONTENTS

ABSTRACT

The project focuses on developing a machine learning-based surveillance system designed to enhance traffic safety by detecting bike riders without helmets and identifying triple riders in real-time. Leveraging advanced computer vision techniques, the system utilizes Convolutional Neural Networks (CNNs) and state-of-the-art detection algorithms such as YOLO (You Only Look Once) and Faster R-CNN to analyse live video feeds from traffic cameras.

With high accuracy (96.66%) and a low false alarm rate of 0.5%, the system reliably identifies violations, even in complex traffic scenarios. It processes video feeds frame by frame, detecting helmet-less riders and triple riding through automated classification and object detection. Integration with pre-trained models like ResNet and EfficientNet further enhances its performance.

The adaptive learning mechanism enables continuous improvement through training on new datasets, ensuring the system remains robust across diverse environments. Designed for seamless integration into existing traffic management systems, this solution offers real-time monitoring, automated detection, and scalable deployment, significantly contributing to safer roadways and improved traffic regulation.

In addition to detecting helmet violations and triple riding, the system can be further enhanced by integrating license plate recognition to automate fine generation. By leveraging Optical Character Recognition (OCR) techniques, the system can extract vehicle registration numbers from detected violators, streamlining law enforcement and reducing the need for manual intervention. This feature enables authorities to issue automated penalty notifications, ensuring a more efficient and transparent traffic monitoring process. Moreover, the system's scalability allows it to be deployed across multiple urban locations with varying traffic densities, adapting to different lighting conditions and environmental factors for consistent performance.

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF THE PROJECT

The project focuses on developing a Machine Learning-Based Surveillance System designed to improve traffic safety by automatically detecting bike riders who are not wearing helmets and identifying instances of triple riding in real-time. With the increasing number of two-wheeled vehicles on the roads, safety violations such as riders not wearing helmets or carrying more passengers than allowed are common causes of accidents. The goal of this project is to create an efficient, automated system that can address these issues without relying on manual intervention..

To achieve this, the system utilizes advanced machine learning algorithms and computer vision techniques. The primary technologies used are Convolutional Neural Networks (CNNs) and object detection models such as YOLO (You Only Look Once) and Faster R-CNN. These models are capable of processing video feeds to identify violations in real-time. The system is designed to analyze live footage from surveillance cameras placed at traffic hotspots or along highways, ensuring comprehensive monitoring of road safety.

By continuously analyzing traffic, the system can quickly detect bike riders without helmets and identify if a vehicle is carrying more than the allowed number of passengers (triple riding). The automated detection process eliminates the need for manual monitoring, reducing the burden on traffic authorities while ensuring consistent enforcement of traffic rules. The system processes video feeds frame by frame, classifying objects and detecting violations with high accuracy.

The system operates with a remarkable accuracy rate of 96.66% in detecting helmet-less riders and has a low false alarm rate of 0.5%, ensuring reliable and precise detection. With the help of adaptive learning, the system continuously improves its performance by collecting new data and retraining the models. This feature allows the system to adapt to different environmental conditions, such as varying lighting and traffic densities, maintaining its efficiency across diverse locations.

Designed for seamless integration with existing traffic management systems, this solution offers a scalable and flexible approach to traffic law enforcement.

1.2 OBJECTIVE OF THE PROJECT

The objective of this project is to develop an advanced machine learning-based surveillance system that can automatically detect bike riders without helmets and identify triple riding in real-time video feeds. The goal is to create a fully automated system capable of monitoring traffic and detecting these violations without requiring manual intervention, significantly improving road safety. With the increasing number of two-wheeled vehicles on the road, particularly in densely populated areas, this system aims to enhance safety by identifying violations related to helmet use and overcrowding on motorcycles.

A key objective of the project is to achieve high accuracy in violation detection, with the system designed to identify helmet-less riders and triple riders with an accuracy of 96.66%. Additionally, it aims to minimize false positives, ensuring the reliability and efficiency of the system. By using machine learning models like YOLO (You Only Look Once) and Faster R-CNN, the system processes real-time video feeds to detect violations quickly, allowing for prompt enforcement actions.

Ultimately, the project aims to provide a cost-effective, efficient, and real-time solution to traffic monitoring, helping authorities enforce traffic laws more effectively and reduce accidents caused by helmet non-compliance and triple riding. By automating the detection process, the system not only saves time and resources but also helps improve the overall safety and regulation of roads, contributing to safer communities.

1.3 ORGANIZATION OF THE PROJECT

The primary objective of this project is to develop a machine learning-based surveillance system that can automatically detect bike riders without helmets and identify triple riding incidents in real-time. By leveraging advanced computer vision techniques, the system processes live video feeds from surveillance cameras installed at key traffic locations. This eliminates the need for manual intervention, allowing for continuous and efficient monitoring of traffic violations. The system ensures real-time operation, enabling authorities to take immediate action and improve overall traffic law enforcement.

To achieve high accuracy and efficiency, the system collects and preprocesses video data using image enhancement techniques before passing it through machine learning algorithms for analysis. The project employs Convolutional Neural Networks (CNNs), YOLO

(You Only Look Once), and Faster R-CNN for real-time detection of helmet-less riders and triple riders. These models analyze video frames, classify objects, and accurately identify violations, ensuring a low false alarm rate. The adaptive learning mechanism further enhances detection accuracy by continuously updating the model with new data.

Once a violation is detected, the system generates real-time alerts that can be transmitted to traffic authorities, displayed on LED screens, or sent as notifications through a mobile application. This immediate feedback mechanism helps enforce traffic laws effectively. Additionally, real-time data logging records crucial details such as the time, location, and type of violation. Traffic authorities can use this data for further analysis, improving law enforcement strategies and traffic management planning.

The system is designed to be highly scalable and seamlessly integrates with existing traffic management infrastructure. It can function alongside other surveillance systems, cameras, and traffic monitoring technologies, making it a versatile solution for enhancing road safety. The cost-effective design, utilizing affordable cameras, sensors, and open-source machine learning models, ensures accessibility and feasibility for deployment in both urban and highway environments.

By automating traffic violation detection, this system contributes significantly to reducing accidents caused by helmet non-compliance and overcrowding on motorcycles. The efficient, data-driven approach streamlines traffic law enforcement and enhances road safety for all commuters. With its advanced detection capabilities, real-time operation, and scalable design, the project serves as a comprehensive solution for improving traffic regulation and ensuring safer roads.

CHAPTER 2

LITERATURE SURVEY

2.1EXISTING SYSTEM

Current methods for monitoring helmet usage and triple riding on two-wheelers rely on a mix of manual enforcement, basic surveillance technologies, and traditional image processing techniques. These approaches, while functional, have several limitations:

- **Manual Inspection by Traffic Officers**:

Traffic officers monitor roads to enforce helmet laws and prevent triple riding. This method is labor-intensive, prone to human error, and limited in coverage, as it cannot effectively monitor large areas or multiple lanes simultaneously.

- **Basic CCTV Camera Systems**

CCTV cameras are installed at strategic locations to capture traffic footage. However, these systems often lack advanced image processing capabilities, requiring manual review of recorded videos to identify violations, which is time-consuming and inefficient.

- **Infrared Cameras**:

Infrared cameras are used to detect helmets based on heat signatures. These systems are limited by environmental factors such as lighting conditions and often struggle to differentiate helmets from other objects.

- **Traditional Image Processing Techniques**:

Basic image processing algorithms are employed to identify helmets and count the number of riders. These methods often lack the sophistication required for real-time detection and are less reliable in crowded or dynamic traffic scenarios.

2.2 PROPOSED SYSTEM

The proposed system leverages machine learning and advanced computer vision techniques to develop an automated surveillance solution capable of detecting bike riders without helmets and identifying triple riders in real-time. Unlike traditional methods, which rely on manual monitoring, this system integrates state-of-the-art object detection algorithms such as YOLO (You Only Look Once) and Faster R-CNN to process live video feeds from surveillance cameras. By analyzing video frames in real-time, the system accurately identifies violations, ensuring effective enforcement of traffic laws without human intervention.

To enhance accuracy and reliability, the system employs Convolutional Neural Networks (CNNs) and pre-trained models like ResNet and EfficientNet, achieving a detection accuracy of 96.66% with a false alarm rate as low as 0.5%. Additionally, the system incorporates adaptive learning mechanisms, allowing it to continuously improve by training on new datasets. This ensures robustness and adaptability to diverse traffic environments, lighting conditions, and varying vehicle densities, making it a scalable and efficient solution.

One of the key strengths of the proposed system is its seamless integration with existing traffic management infrastructure. The system is designed to handle multiple camera inputs and can be deployed across high-traffic zones for broader coverage. Automated reporting further enhances its efficiency by flagging violations and generating alerts for traffic authorities, reducing the dependency on manual monitoring. This automation enables a streamlined approach to law enforcement, ensuring prompt action against violations.

The system offers several advantages, including real-time operation for immediate violation detection, high scalability to monitor large areas, and enhanced accuracy to minimize false detections. Additionally, its automation reduces the need for human operators, improving efficiency, while its versatility ensures reliable performance under varied lighting and weather conditions. By integrating cutting-edge machine learning algorithms with adaptive technologies, the proposed system presents a highly reliable, efficient, and scalable solution for improving traffic safety and compliance.

2.3 EMBEDDED INTRODUCTION

An embedded system is a combination of computer hardware and software designed for a specific function or functions within a larger system. The systems can be programmable or with fixed functionality. Industrial machines, consumer electronics, agricultural and process industry devices, automobiles, medical equipment, cameras, household appliances, airplanes, vending machines and toys, as well as mobile devices, are possible locations for an embedded system.

While embedded systems are computing systems, they can range from having no user interface (UI) -- for example, on devices in which the system is designed to perform a single task -- to complex graphical user interfaces (GUIs), such as in mobile devices. User interfaces can

include buttons, LEDs and touchscreen sensing. Some systems use remote user interfaces as well.

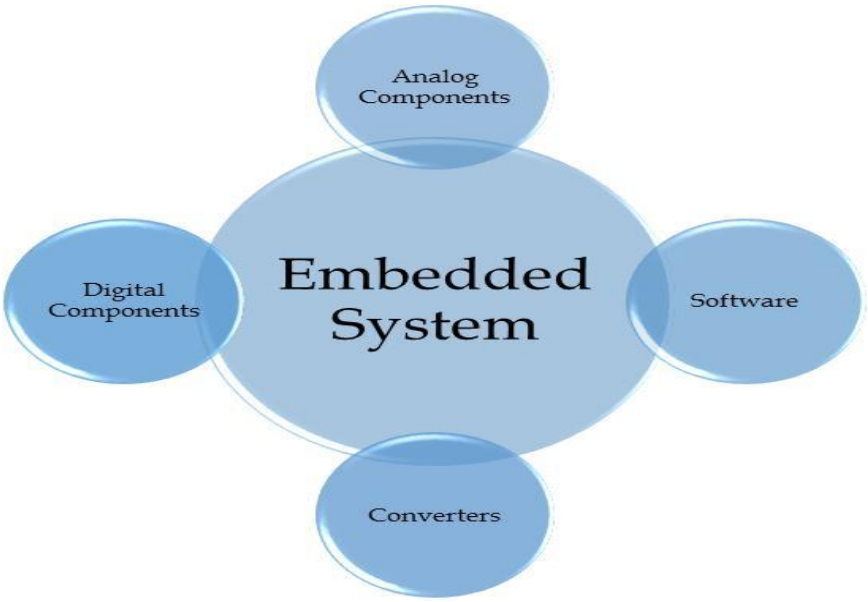

Fig:2.2 EMBEDDED SYSTEM

History of embedded systems

Embedded systems date back to the 1960s. Charles Stark Draper developed an integrated circuit (IC) in 1961 to reduce the size and weight of the Apollo Guidance Computer, the digital system installed on the Apollo Command Module and Lunar Module. The first computer to use ICs, it helped astronauts collect real-time flight data.

In 1965, Autonoetic, now a part of Boeing, developed the D-17B, the computer used in the Minuteman I missile guidance system. It is widely recognized as the first mass-produced embedded system. When the Minuteman II went into production in 1966, the D-17B was replaced with the NS-17 missile guidance system, known for its high-volume use of integrated circuits.

In 1968, the first embedded system for a vehicle was released; the Volkswagen 1600 used a microprocessor to control its electronic fuel injection system.

By the late 1960s and early 1970s, the price of integrated circuits dropped, and usage surged. The first microcontroller was developed by Texas Instruments in 1971. The TMS 1000 series, which became commercially available in 1974, contained a 4-bit processor, read-only memory (ROM) and random-access memory (RAM), and cost around $2 apiece in bulk orders.

Also, in 1971, Intel released what is widely recognized as the first commercially available processor, the 4004. The 4-bit microprocessor was designed for use in calculators and

small electronics, though it required eternal memory and support chips. The 8-bit Intel 8008, released in 1972, had 16 KB of memory; the Intel 8080 followed in 1974 with 64 KB of memory. The 8080's successor, x86 series, was released in 1978 and is still largely in use today.

In 1987, the first embedded operating system, the real-time VxWorks, was released by Wind River, followed by Microsoft's Windows Embedded CE in 1996. By the late 1990s, the first embedded Linux products began to appear. Today, Linux is used in almost all embedded devices.

Characteristics of embedded systems

The main characteristic of embedded systems is that they are task specific. They perform a single task within a larger system. For example, a mobile phone is not an embedded system, it is a combination of embedded systems that together allow it to perform a variety of general-purpose tasks. The embedded systems within it perform specialized functions. For example, the GUI performs the singular function of allowing the user to interface with the device. In short, they are programmable computers, but designed for specific purposes, not general ones.

The hardware of embedded systems is based around microprocessors and microcontrollers. Microprocessors are very similar to microcontrollers, and generally refer to

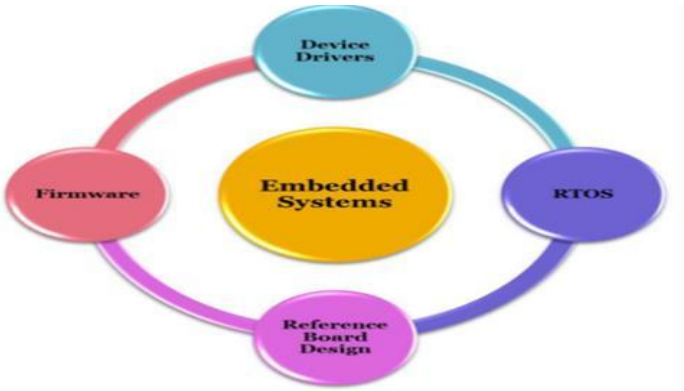

Fig:2.3 CHARACTERISTICS OF EMBEDDED SYSTEMS

Additionally, embedded systems can include the following characteristics:

- comprised of hardware, software and firmware;
- embedded in a larger system to perform a specific function as they are built for specialized tasks within the system, not various tasks;
- either microprocessor-based or microcontroller-based -- both are integrated circuits that give the system compute power;

- often used for sensing and real-time computing in internet of things (IoT) devices -- devices that are internet-connected and do not require a user to operate;
- vary in complexity and in function, which affects the type of software, firmware and hardware they use; and
- often required to perform their function under a time constraint to keep the larger system functioning properly.

Embedded systems vary in complexity, but generally consist of three main elements:

- **Hardware.** The hardware of embedded systems is based around microprocessors and microcontrollers. Microprocessors are very similar to microcontrollers, and generally refer to a CPU that is integrated with other basic computing components such as memory chips and digital signal processors (DSP). Microcontrollers have those components built into one chip.
- **Software.** Software for embedded systems can vary in complexity. However, industrial grade microcontrollers and embedded IoT systems generally run very simple software that requires little memory.

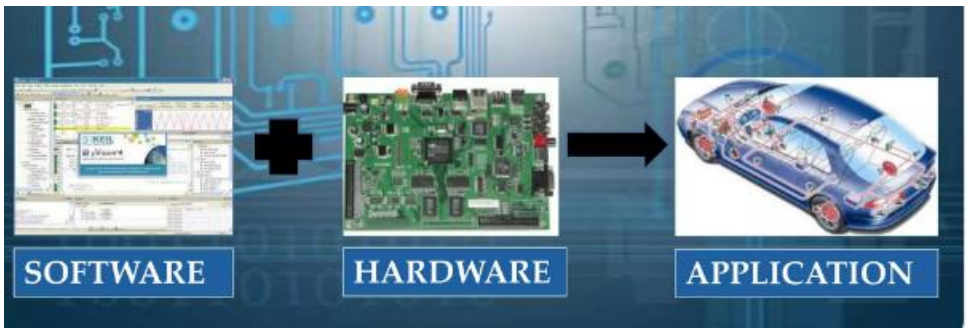

Fig:2.4 BLOCKS OF EMBEDDED SYSTEMS

- **Firmware**. Embedded firmware is usually used in more complex embedded systems to connect the software to the hardware. Firmware is the software that interfaces directly with the hardware. A simpler system may just have software directly in the chip, but more complicated systems need firmware under more complex software applications and operating systems.

2.4 WHY EMBEDDED?

An embedded system is a computer system with a particular defined function within a larger mechanical or electrical system. They control many devices in common use. They consume low power, are of a small size and their cost is low per-unit.

Modern embedded systems are often based on micro-controllers. A micro-controller is a small computer on a single integrated circuit which contains a processor core, memory, and programmable input and output peripherals. As Embedded system is dedicated to perform specific tasks therefore, they can be optimized to reduce the size and cost of the product and increase the reliability and performance.

Almost every Electronic Gadget around us is an Embedded System, digital watches, MP3 players, Washing Machine, Security System, scanner, printer, a cellular phone, Elevators, ATM, Vendor Machines, GPS, traffic lights, Remote Control, Microwave Oven and many more. The uses of embedded systems are virtually limitless because every day new products are introduced to the market which utilize embedded computers in a number of ways.

Embedded Systems has brought about a revolution in science. It is also a part of an Internet of Things (IoT) – a technology in which objects, animals or people are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

Fig:2.5 EMBEDDED SYSTEMS HARDWARE

Let's make it easy for you. For Example – You are sitting in a train headed to your destination and you are already fifty miles away from your home and suddenly you realise that you forgot to switch of the fan. Not to worry, you can switch it off just by clicking a button on your cell phone using this technology – The Internet of Things.

Well, this is just one good thing about IoT. We can monitor Pollution Levels, we can control the intensity of street lights as per the season and weather requirements, IoT can also provide the parents with real-time information about their baby's breathing, skin temperature, body position, and activity level on their smartphones and many other applications which can make our life easy.

2.5 DESIGN APPROACHES

A system designed with the embedding of hardware and software together for a specific function with a larger area is embedded system design. In embedded system design, a microcontroller plays a vital role. Micro-controller is based on Harvard architecture, it is an important component of an embedded system. External processor, internal memory and i/o components are interfaced with the microcontroller. It occupies less area, less power consumption. The application of microcontrollers is MP3, washing machines.

Critical Embedded Systems (CES) are systems in which failures are potentiallycatastrophic and, therefore, hard constraints are imposed on them. In the last years the amount of software accommodated within CES has considerably changed.

For example, in smart cars the amount of software has grown about 100 times compared to previous years. This change means that software design for these systems is also bounded to hard constraints (e.g., high security and performance). Along the evolution of CES, the approaches for designing them are also changing rapidly, so as to fit the specialized needs of CES. Thus, a broad understanding of such approaches is missing.

Steps in the Embedded System Design Process

The different steps in the embedded system design flow/flow diagram include the following.s

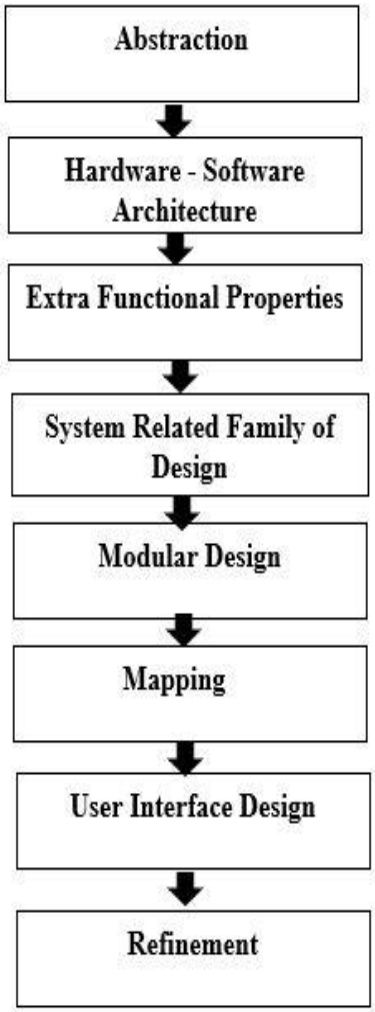

Fig:2.6 EMBEDDED DESIGN-PROCESS-STEPS

Abstraction

In this stage the problem related to the system is abstracted.

Hardware – Software Architecture

Proper knowledge of hardware and software to be known before starting any design process.

Extra Functional Properties

Extra functions to be implemented are to be understood completely from the main design.

System Related Family of Design

When designing a system, one should refer to a previous system-related family of design.

Modular Design

Separate module designs must be made so that they can be used later on when required.

Mapping

Based on software mapping is done. For example, data flow and program flow are mapped into one.

User Interface Design

In user interface design it depends on user requirements, environment analysis and function of the system. For example, on a mobile phone if we want to reduce the power consumption of mobile phones, we take care of other parameters, so that power consumption can be reduced.

Refinement

Every component and module must be refined appropriately so that the software team can understand.

Architectural description language is used to describe the software design.

- Control Hierarchy
- Partition of structure
- Data structure and hierarchy
- Software Procedure.

In user interface design it depends on user requirements, environment analysis and function of the system. For example, on a mobile phone if we want to reduce the power consumption of mobile phones, we take care of other parameters, so that power consumption can be reduced. To help countries and health-care facilities to achieve system change and adopt alcohol-based hand rubs as the gold standard for hand hygiene in health care, WHO has identified formulations for their local preparation. Logistic, economic, safety, and cultural.

Table:1.1 DESIGN PARAMETERS OF AN EMBEDDED SYSTEM

Design Parameters of an Embedded System	Function
Power Dissipation	Always maintained low
Performance	Should be high
Process Deadlines	The process/task should be completed within a specified time.
Manufacturing Cost	Should be maintained.
Engineering Cost	It is the cost for the edit-test-debug of hardware and software.
Size	Size is defined in terms of memory RAM/ROM/Flash Memory/Physical Memory.
Prototype	It is the total time taken for developing a system and testing it.
Safety	System safety should be taken like phone locking, user safety like engine breaks down safety measure must be taken.
Maintenance	Proper maintenance of the system must be taken, in order to avoid system failure.
Time to market	It is the time taken for the product /system developed to be launched into the market.

2.5.1 SPECIFICATION

During this part of the design process, the informal requirements of the analysis are transformed to formal specification using SDL.

2.5.2 SYSTEM-SYNTHESIS

For performing an automatic HW/SW partitioning, the system synthesis step translates the SDL specification to an internal system model switch contains problem graph& architecture graph. After system synthesis, the resulting system model is translated back to SDL.

2.5.3 IMPLEMENTATION-SYNTHESIS

On a prototyping platform, the implementation of the system under development is executed with the software parts running on multiprocessor unit and the hardware part running on a FPGA board known as phoenix, prototype hardware for Embedded Network Interconnect Accelerators

2.5.4 APPLICATIONS

Embedded systems are finding their way into robotic toys and electronic pets, intelligent cars and remote controllable home appliances. All the major toy makers across the world have been coming out with advanced interactive toys that can become our friends for life. 'Furby' and 'AIBO' are good examples at this kind. Furbies have a distinct life cycle just like human beings, starting from being a baby and growing to an adult one. In AIBO first two letters stand for Artificial Intelligence. Next two letters represent robot.

The AIBO is robotic dog. Embedded systems in cars also known as Telematic Systems are used to provide navigational security communication &entertainment services using GPS, satellite. Home appliances are going the embedded way. LG electronics digital DIOS refrigerator can be used for surfing the net, checking e-mail, making video phone calls and watching TV.IBM is developing an air conditioner that we can control over the net. Embedded systems cover such a broad range of products that generalization is difficult. Here are some broad categories.

In the automobile sector, embedded systems, often referred to as Telematic Systems, are essential for navigation, safety, entertainment, and autonomous driving. Modern cars feature GPS-based navigation, collision detection, adaptive cruise control, and parking assistance, all powered by real-time data processing and AI algorithms.

Companies like Tesla use advanced embedded systems and LIDAR sensors to enable self-driving capabilities, improving road safety and driving efficiency.

- **Aerospace and defence electronics**: Fire control, radar, robotics/sensors, sonar.
- **Automotive**: Autobody electronics, auto power train, auto safety, car information systems.
- **Broadcast & entertainment**: Analog and digital sound products, camaras, DVDs, Set top boxes, virtual reality systems, graphic products.
- **Consumer/internet appliances**: Business handheld computers, business network computers/terminals, electronic books, internet smart handheld devices, PDAs.
- **Data communications:** Analog modems, ATM switches, cable modems, XDSL modems, Ethernet switches, concentrators.
- **Digital imaging**: Copiers, digital still cameras, Fax machines, printers, scanners.

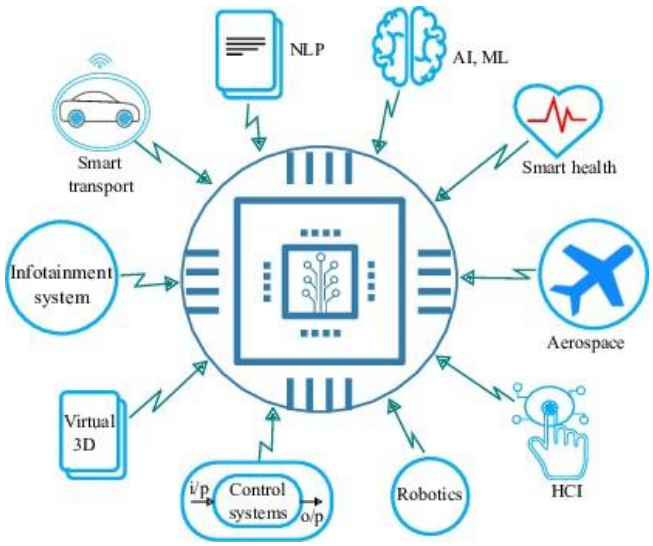

Fig:2.7 APPLICATIONS OF EMBEDDED SYSTEMS

- **Healthcare:** Automated insulin pumps, ECG monitors & heart rate sensors, MRI & CT scanners, Smart prosthetics, Wireless patient monitoring
- **Industrial measurement and control:** Hydro electric utility research & management traffic management systems, train marine vessel management systems.
- **Medical electronics:** Diagnostic devices, real time medical imaging systems, surgical devices, critical care systems.
- **Server I/O:** Embedded servers, enterprise PC servers, PCI LAN/NIC controllers, RAID devices, SCSI devices.
-
- **Telecommunications**: ATM communication products, base stations, networking

switches, SONET/SDH cross connect, multiplexer.

- **Networking devices**: Routers, modems, and IoT gateways enable seamless internet connectivity, data transmission, and smart device communication in embedded systems.
- **Virtual Reality (VR) & Augmented Reality (AR):** Virtual Reality (VR) immerses users in a fully digital environment, while Augmented Reality (AR) overlays digital elements onto the real world, enhancing the user's perception of their surroundings.

2.5.5 FEATURES

Embedded systems are specialized computing devices designed for dedicated tasks with reliability and efficiency. They operate under real-time constraints, using minimal memory and low power consumption to ensure stable performance. Their task-specific design emphasizes fault tolerance, low cost, and minimal interface requirements.

By focusing on optimized functionality, embedded systems deliver high efficiency in processing and managing data. They are widely used in automotive, healthcare, industrial automation, and IoT applications. With compact designs and user-friendly operation, these systems ensure high stability and consistent performance, meeting strict timing requirements and enabling seamless, reliable operation in diverse environments. They continuously excel in efficiency.

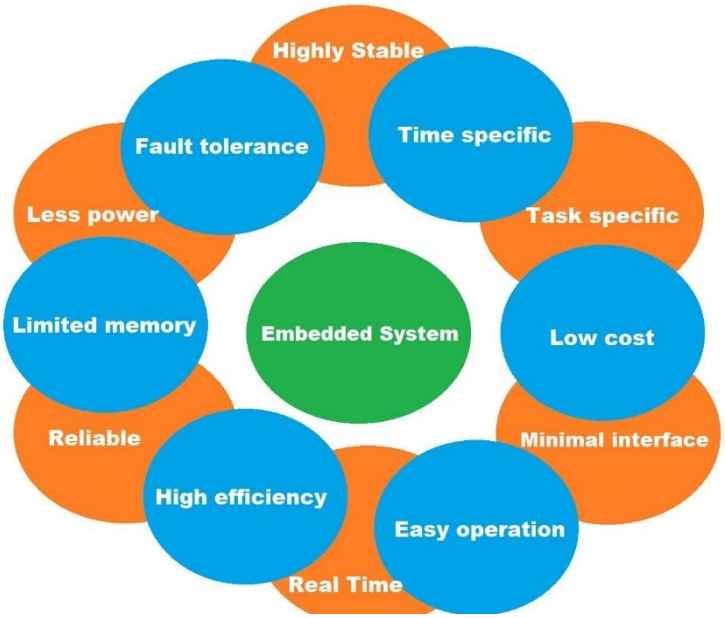

Fig: 2.8 FEATURES OF EMBEDDED SYSTEM

- **Highly Stable** – Ensures reliable performance over long periods.
- **Time-Specific** – Operates within strict time constraints.
- **Task-Specific** – Designed for a dedicated function.
- **Low Cost** – Cost-effective due to optimized hardware and software.
- **Minimal Interface** – Requires simple user interaction.
- **Easy Operation** – User-friendly and automated functionality.
- **Real-Time Processing** – Responds quickly to inputs for real-time applications.
- **High Efficiency** – Optimized for maximum performance with limited resources.
- **Reliable** – Functions consistently without frequent failures.
- **Limited Memory** – Operates with minimal RAM and storage.
- **Less Power Consumption** – Designed to use minimal energy.
- **Fault Tolerance** – Can handle errors and failures effectively.

CHAPTER 3

HARDWARE REQUIREMENTS

3.1 HARDWARE

3.1.1 INTRODUCTION TO ARDUINO

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analogue inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button.

It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter. "Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions.

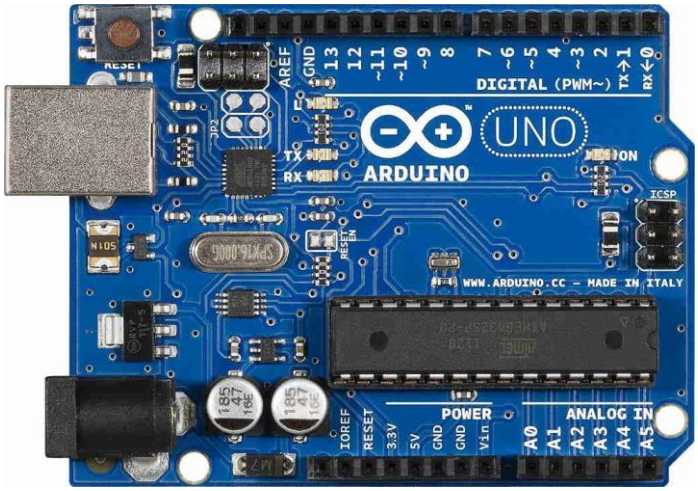

Fig:3.1 ARDUINO UNO

It communicates with external devices via I2C, SPI, and UART protocols, enabling seamless integration with sensors, motors, displays, and other peripherals. The reset button allows users to restart the board without disconnecting power, aiding in program debugging and execution testing. Its open-source nature allows a vast community of developers to contribute libraries and projects, expanding its capabilities. Common applications include automated systems,

home automation, sensor-based projects, robotics, and industrial monitoring. The board's compact size (68.6 mm × 53.4 mm) and lightweight design make it suitable for portable applications.

3.1.2 INTRODUCTION TO ESP32-CAM

The ESP32-CAM is a low-cost, compact camera module based on the ESP32 microcontroller, designed for wireless image processing and video streaming applications. It is widely used in IoT, security systems, and AI-based vision projects due to its built-in Wi-Fi and Bluetooth capabilities. The module comes equipped with a 2MP OV2640 camera, which allows it to capture high-resolution images and stream live video, making it an excellent choice for real-time monitoring applications.

Fig: 3.2 ESP32-CAM

One of the key advantages of ESP32-CAM is its low power consumption and small form factor, making it ideal for embedded systems and remote surveillance. It can be powered through 3.3V to 5V, making it compatible with various microcontroller setups. Additionally, it supports MicroSD card storage, allowing for local data storage, which is beneficial for recording images and videos in applications where continuous internet connectivity is not available.

The ESP32-CAM is extensively used in applications such as face recognition, number plate detection, home automation, and traffic surveillance. Its ability to process images locally reduces the need for external computing resources, making it an efficient and cost-effective

solution. Furthermore, with its built-in wireless connectivity, it can transmit images and video to cloud servers or other devices for further processing.

In the Non-Helmet Rider Detection System, the ESP32-CAM plays a crucial role in capturing real-time images and video footage of motorcycles on the road. These images are then processed using YOLO-based object detection to check for helmet usage. If a violation is detected, the system extracts the license plate number using OCR for further action. Its seamless integration with AI and deep learning models makes it an ideal choice for smart surveillance and law enforcement applications**.**

3.1.3 INTRODUCTION TO BLUETOOTH MODULE HC-05

The HC-05 Bluetooth module is a widely used, low-cost, and easy-to-interface wireless communication module designed for short-range data transmission. It operates on Bluetooth 2.0+EDR (Enhanced Data Rate) technology, allowing devices to communicate wirelessly over a range of 10 to 100 meters, depending on environmental conditions. The module supports both master and slave modes, enabling it to connect with microcontrollers, smartphones, or other Bluetooth-enabled devices for seamless communication.

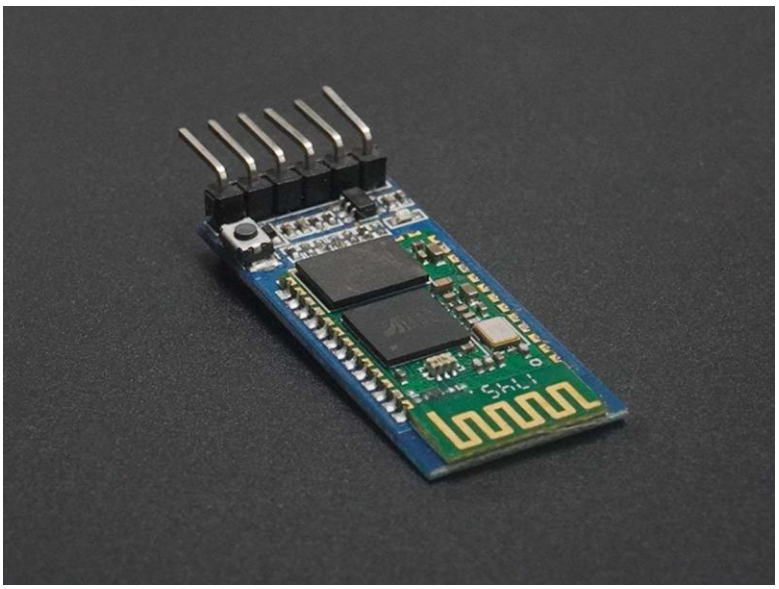

FIG:3.3 Bluetooth Module HC-05

One of the main advantages of the HC-05 module is its serial communication (UART) interface, which allows it to be easily integrated with microcontrollers like Arduino, ESP32, and Raspberry Pi. It operates at a default baud rate of 9600 bps, ensuring stable and efficient data transfer. Additionally, it supports AT commands, which allow users to configure its settings, such as name, password, and mode of operation, making it highly versatile.

The HC-05 module is commonly used in applications such as home automation, wireless robotics, health monitoring systems, and IoT projects. It provides a reliable and energy-efficient solution for data exchange between embedded systems and Bluetooth-enabled devices. Due to its low power consumption, it is also suitable for battery-operated projects where energy efficiency is critical.

In the Non-Helmet Rider Detection System, the HC-05 Bluetooth module can be used to wirelessly transmit helmet violation data from the detection unit to a nearby processing system, smartphone, or law enforcement device. This eliminates the need for wired connections, making the system more flexible, portable, and scalable. By integrating Bluetooth connectivity, the project gains an efficient, real-time communication system, enhancing its usability in traffic monitoring and enforcement applications.

3.1.4 INTRODUCTION TO BLUETOOTH MODULE HC-05

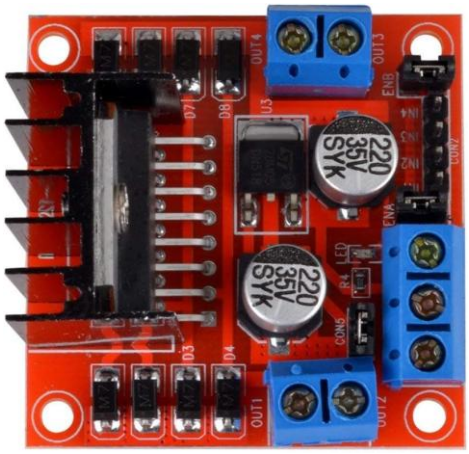

FIG:3.4 L298N Motor Driver

The L298N Motor Driver is a dual H-Bridge motor driver module designed to control the speed and direction of DC motors and stepper motors. It is commonly used in robotics, automation, and embedded systems due to its ability to drive high-power motors efficiently. The module operates with an input voltage range of 5V to 35V and can deliver a continuous current of up to 2A per channel, making it suitable for various motor control applications.

One of the key features of the L298N module is its built-in heat sink, which helps dissipate heat and ensures stable performance during extended operations. It supports PWM (Pulse Width Modulation) control, allowing precise speed regulation of connected motors. The module also

includes two enable pins for activating or deactivating each motor independently, providing greater flexibility in motor control.

In the Non-Helmet Rider Detection System, the L298N Motor Driver can be integrated to control the movement of servo-based barriers or automated gates. When a rider without a helmet is detected, the motor driver can activate a motorized barrier to restrict access or operate an alert mechanism, such as rotating a camera or triggering a visual signal. Additionally, it can be used for camera positioning adjustments, ensuring the best possible angle for helmet and license plate detection.

By adding the L298N Motor Driver, the system gains enhanced motor control capabilities, enabling automated physical responses to traffic violations, improving enforcement efficiency, and making the project more interactive and effective.

CHAPTER 4

SOFTWARE REQUIREMENTS

4.1 PYTHON SOFTWARE

FIG:4.1 Installation of python

The following window shows all the optional features. All the features need to be installed and are checked by default; we need to click next to continue.

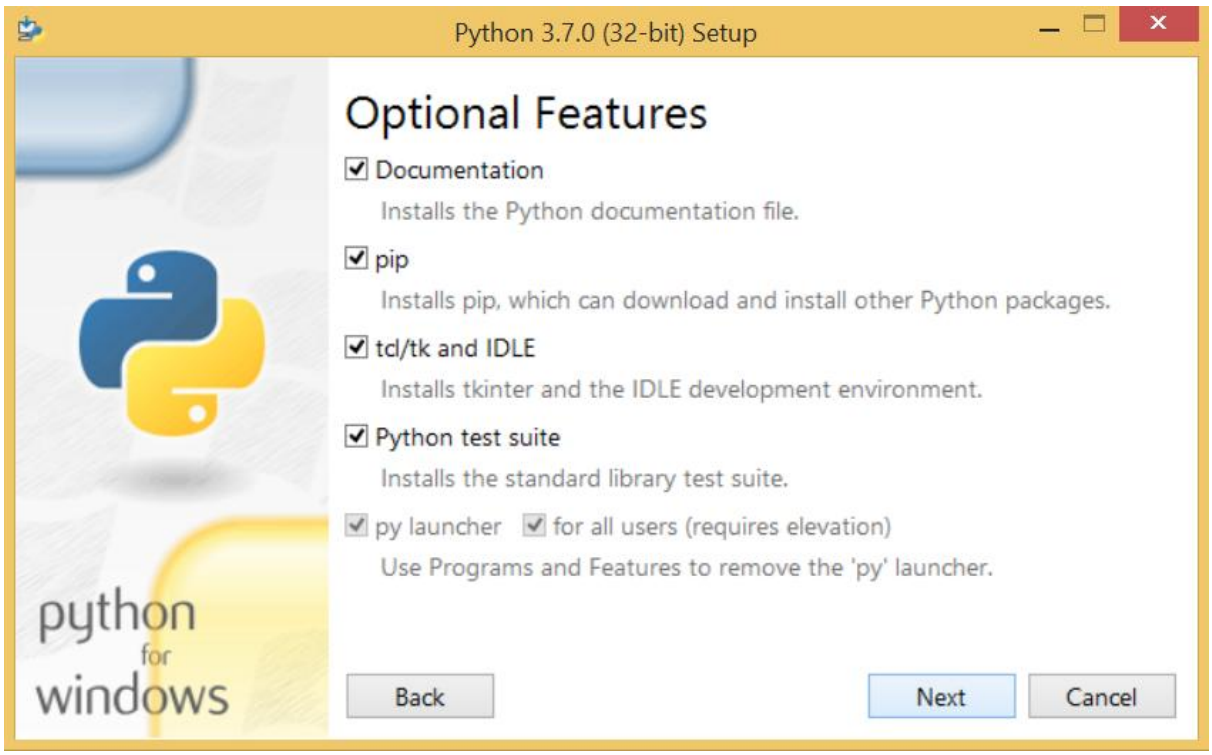

FIG:4.2 Optional features

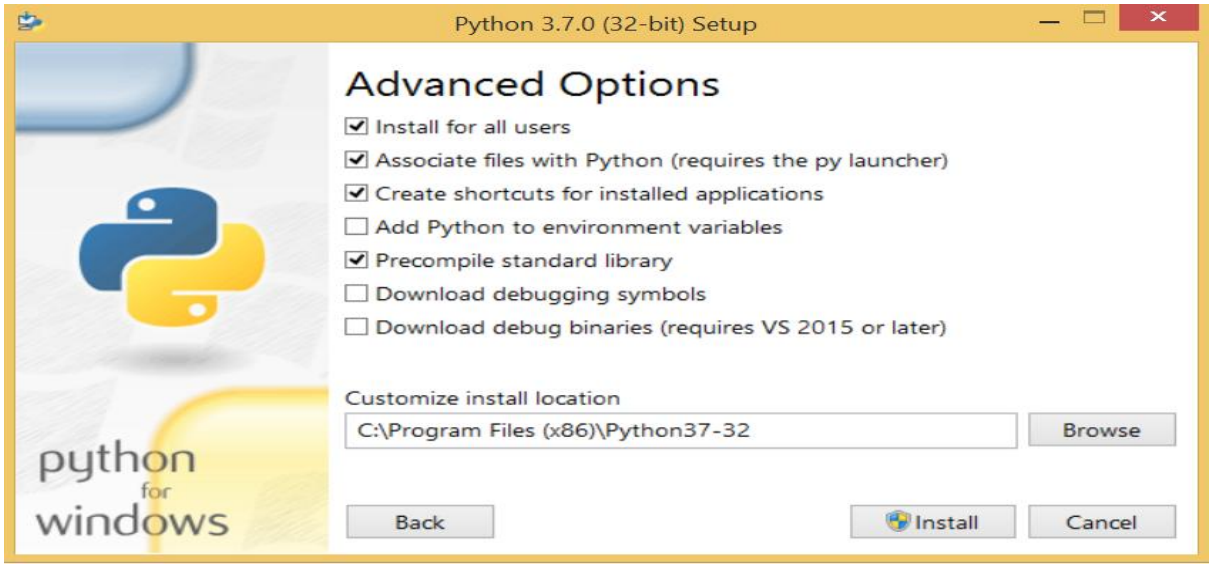

FIG:4.3 Advanced Options

Now, we are ready to install python-3.6.7. Let's install it.

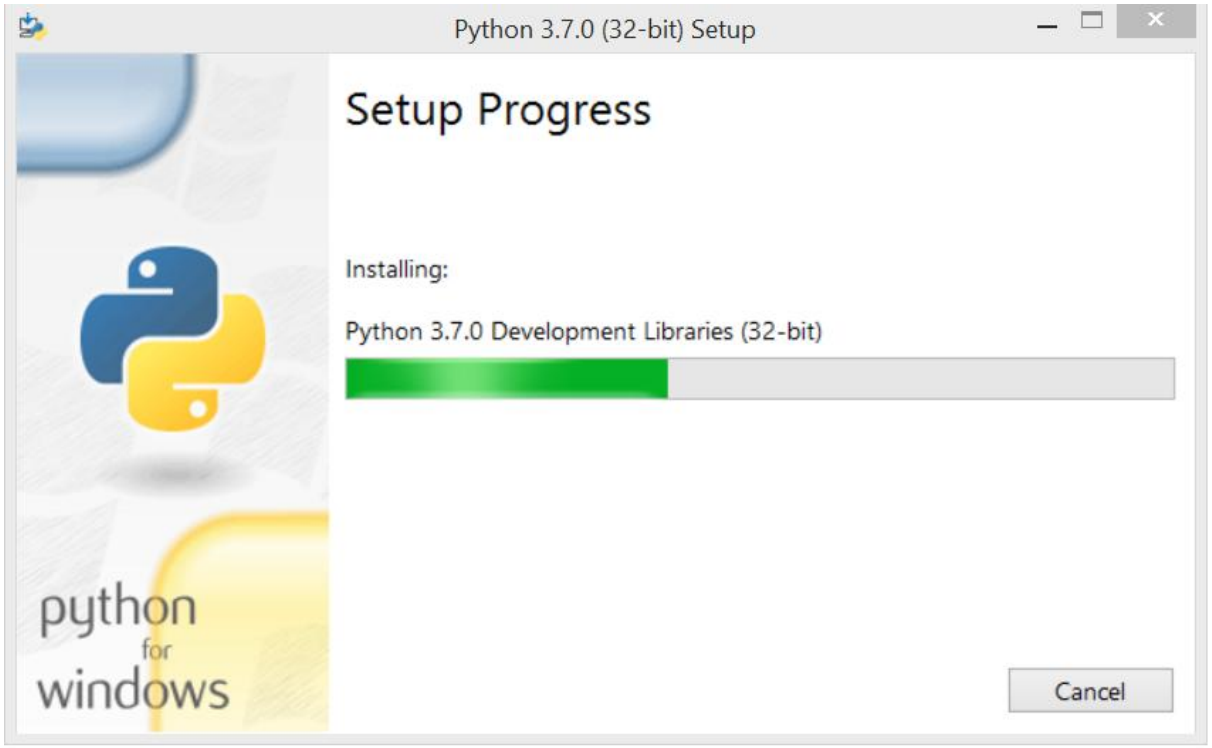

FIG: 4.4 Python setup Progress

Now, try to run python on the command prompt. Type the command **python** in case of python2 or python3 in case of **python3**. It will show an error as given in the below image. It is because we haven't set the path.

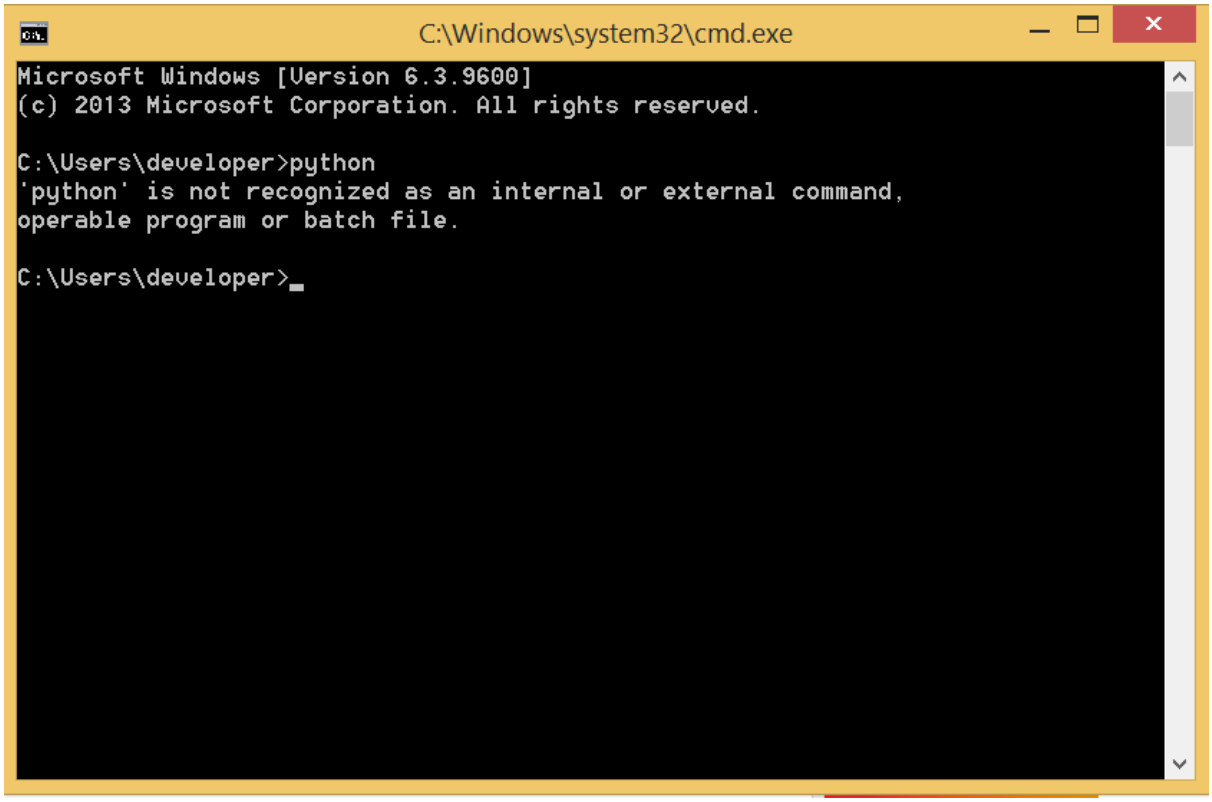

FIG:4.5 Command promt

To set the path of python, we need to the right click on "my computer" and go to Properties → Advanced → Environment Variables.

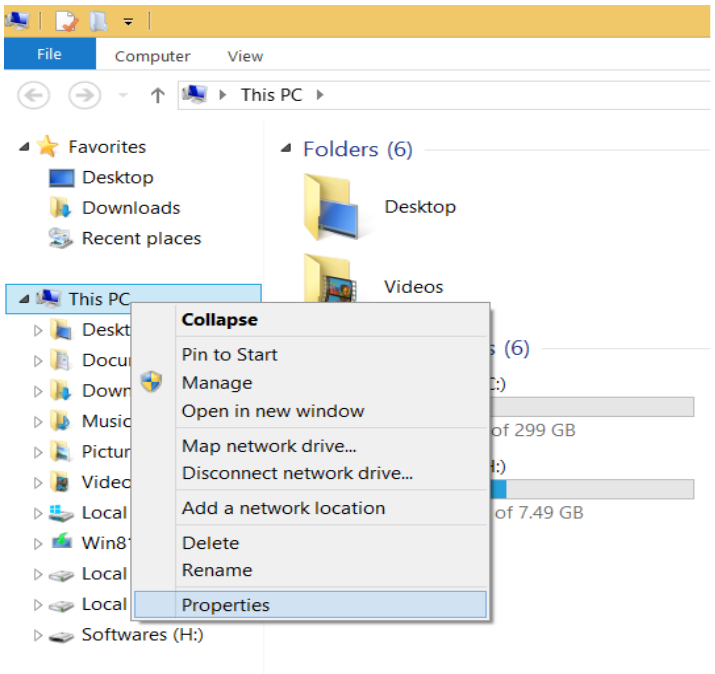

FIG:4.6 Choose Properties

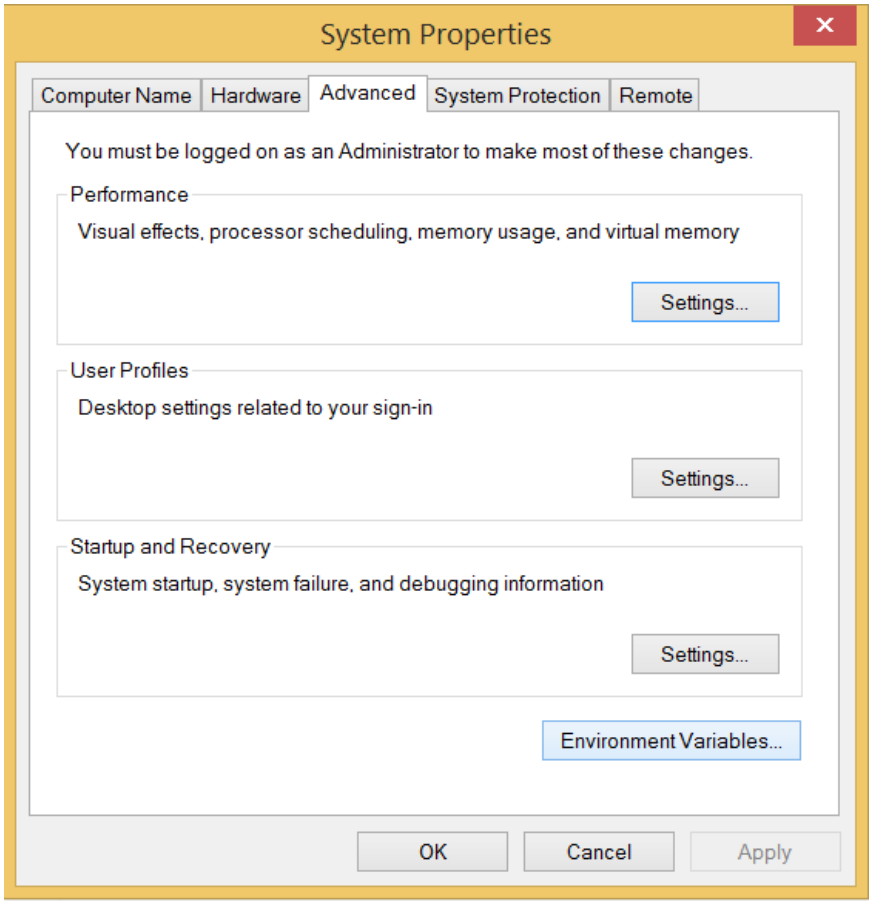

FIG:4.7 System Properties

Add the new path variable in the user variable section.

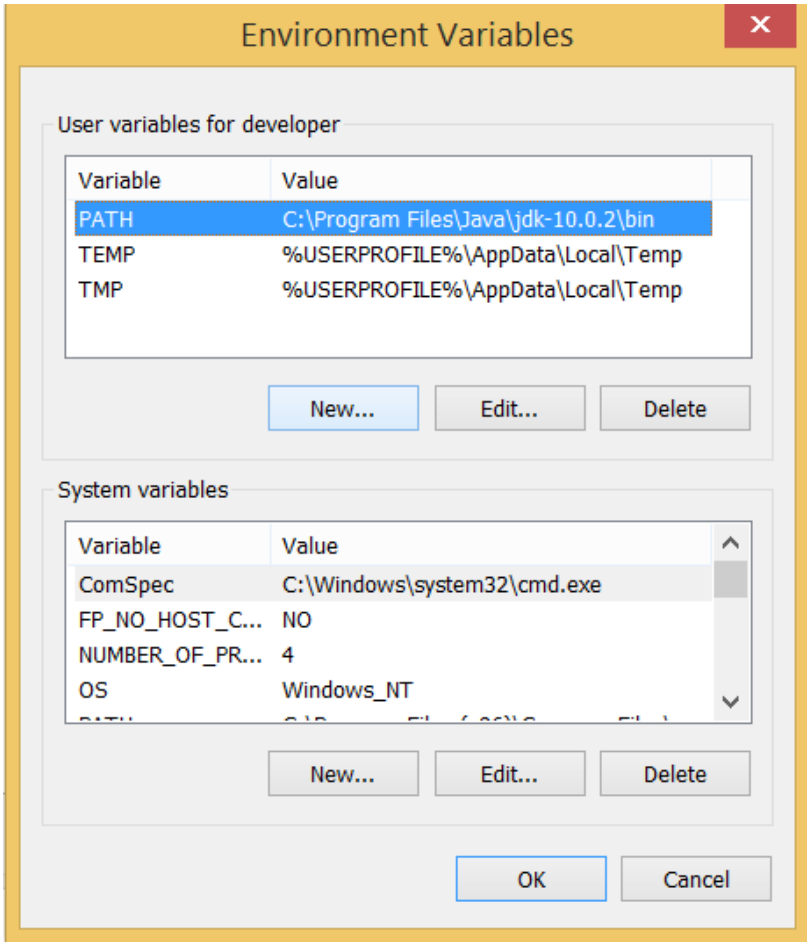

FIG:4.8 Environment Variables

Type **PATH** as the variable name and set the path to the installation directory of the python shown in the below image.

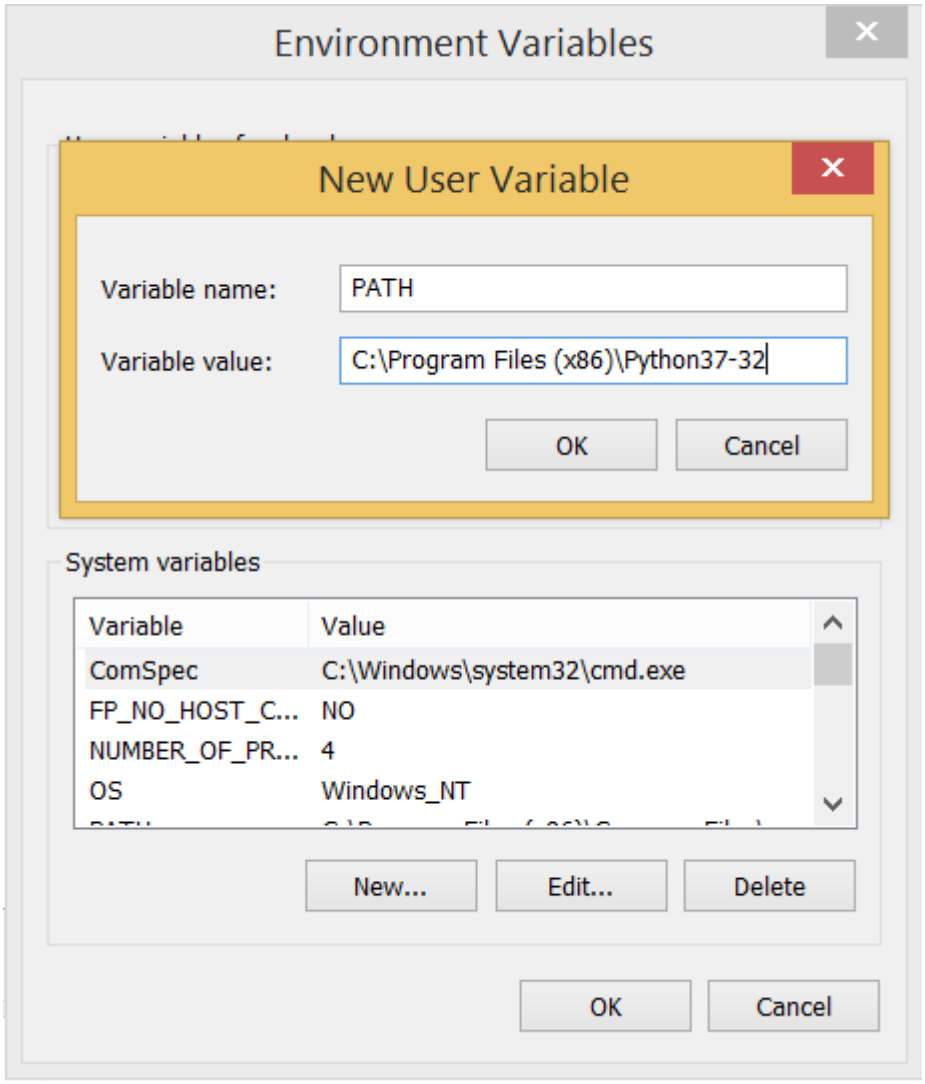

FIG:4.9 New user Variable

Now, the path is set, we are ready to run python on our local system. Restart CMD, and type **python** again. It will open the python interpreter shell where we can execute the python statements.

4.2 ARDUNIO SOFTWARE

APIs and drivers

Software making extensive use of special hardware devices, like high-end display adapters, needs special API or newer device drivers. A good example is DirectX, which is a collection of APIs for handling tasks related to multimedia, especially game programming, on Microsoft platforms

How to Download Arduino IDE

- You can download the Software from Arduino main website. As I said earlier, the software is available for common operating systems like Linux, Windows, and MAX, so make sure you are downloading the correct software version that is easily compatible with your operating system.
- If you aim to download Windows app version, make sure you have Windows 8.1 or Windows 10, as app version is not compatible with Windows 7 or older version of this operating system.
- You can download the latest version of Arduino IDE for Windows (Non-Admin standalone version)

The IDE environment is mainly distributed into three sections

1. Menu Bar
2. Text Editor
3. Output Panel

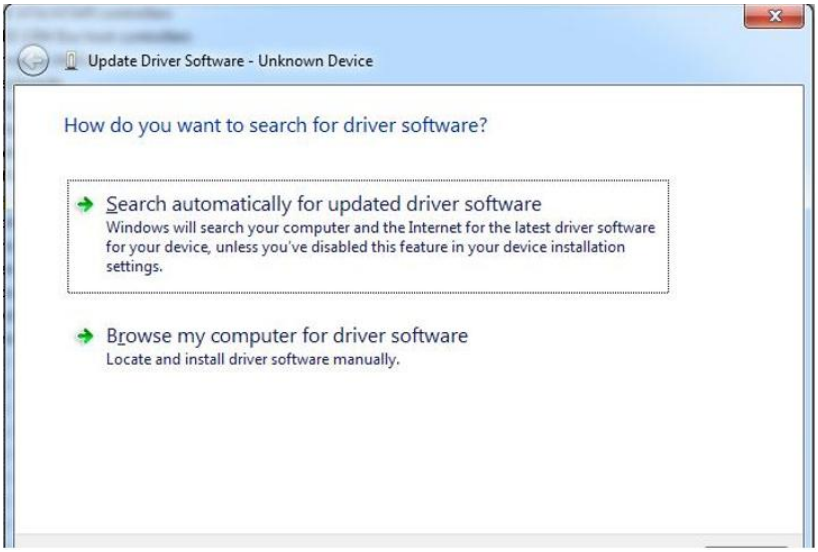

Fig :4.10 DRIVER SELECTION

- Plug in your board and wait for Windows to begin its driver installation process After a few moments, the process will fail, despite its best efforts
- Click on the Start Menu, and open up the Control Panel
- While in the Control Panel, navigate to System and Security. Next, click on System Once the System window is up, open the Device Manager
- Look under Ports (COM & LPT). You should see an open port named “Arduino UNO (COM)”.

- If there is no COM & LPT section, look under ‘Other Devices’ for ‘Unknown Device’
- Right click on the “Arduino UNO (COM)” or “Unknown Device” port and choose the “Update Driver Software” Opti Next, choose the “Browse my computer for Driver software” option.

LAUNCH AND BLINK!

After following the appropriate steps for your software install, we are now ready to test your first program with your Arduino board!

- Launch the Arduino application
- If you disconnected your board, plug it back in
- Open the Blink example sketch by going to: File > Examples > 1.Basics > Blink

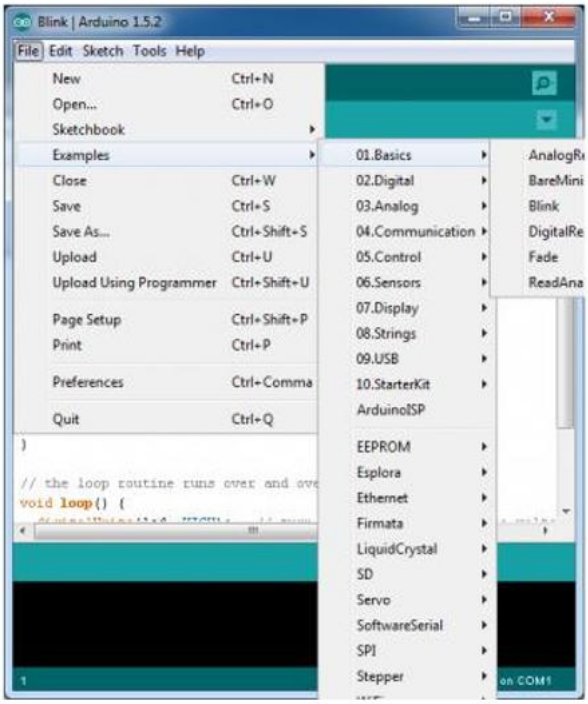

Fig:4.11 SAMPLR PROGRAM

- Select the type of Arduino board you’re using: Tools > Board > your board type

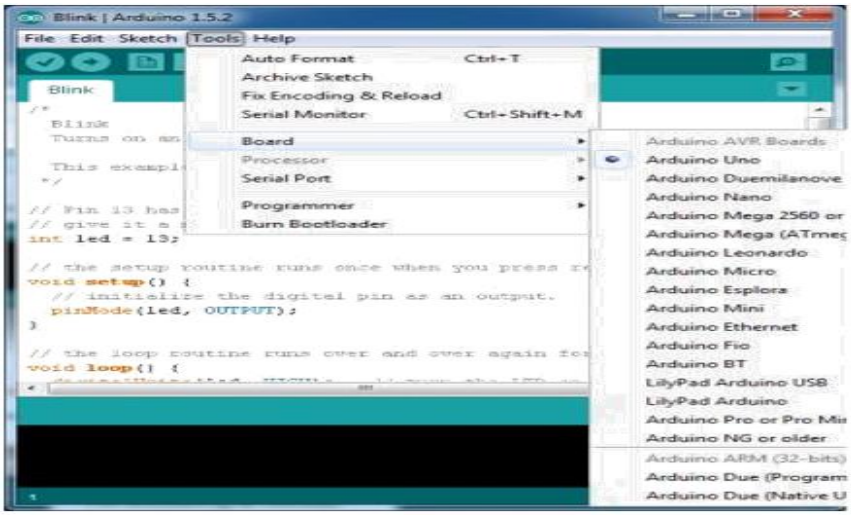

Fig :4.12 BOARD SELECTION

- Select the serial/COM port that your Arduino is attached to: Tools > Port > COM

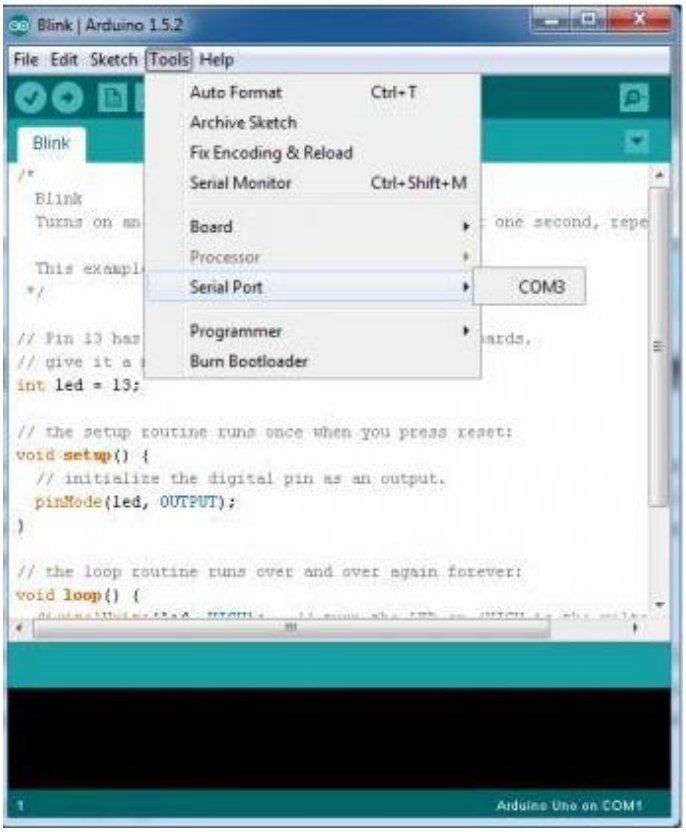

Fig :4.13 PORT SELECTION

CHAPTER 5

WORKING MODEL AND COMPONENTS

5.1 BLOCK DIAGRAM

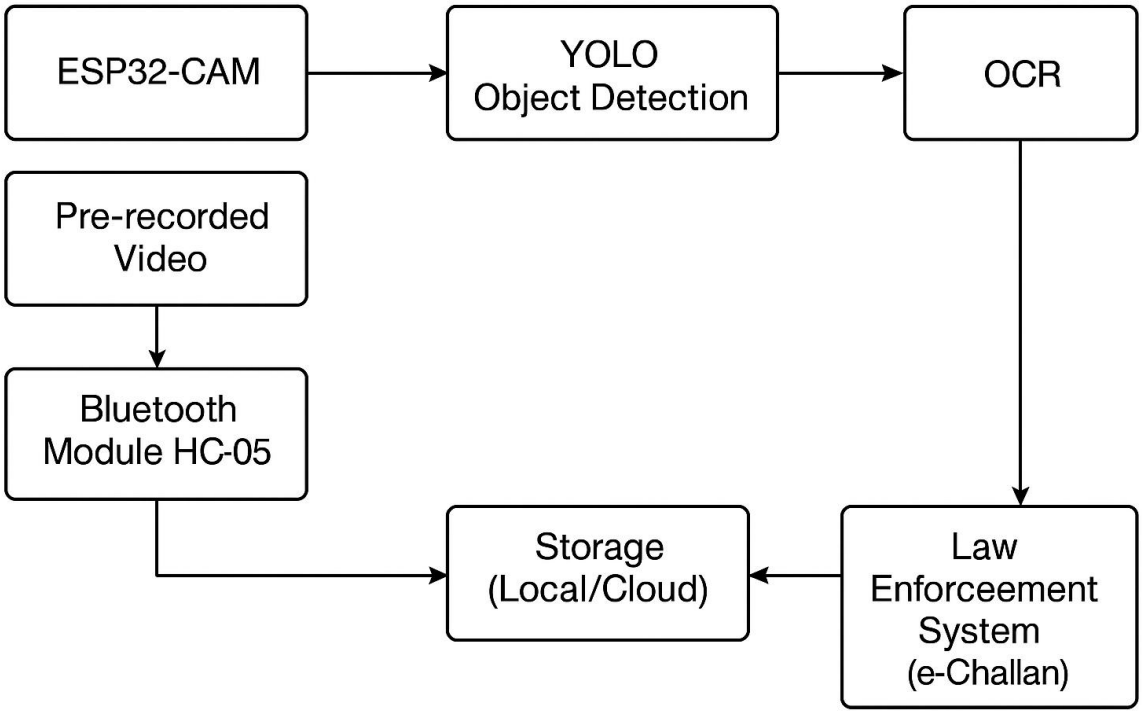

Fig:5.1 BLOCK DIAGRAM

5.2 WORKING

The Non-Helmet Rider Detection System is designed to identify motorcyclists who are not wearing helmets and extract their license plate number for further action. The system utilizes ESP32-CAM, Bluetooth Module HC-05, YOLO-based object detection, and OCR (Optical Character Recognition) to achieve its objectives. The working of the system is explained in the following steps:

1. Video Capture

The system takes input in the form of a real-time video feed from the ESP32-CAM module or a pre-recorded video file. The ESP32-CAM continuously captures frames of the traffic, focusing on motorcycles and their riders.

2. Object Detection using YOLO

Each video frame is processed using the YOLO (You Only Look Once) object detection model to identify key objects such as:

- **Motorcycle** (to confirm the presence of a two-wheeler).
- **Person** (to identify the rider on the motorcycle).
- **Helmet** (to check whether the rider is wearing a helmet).
- **License Plate** (to extract the number if a violation is detected).

If a rider is detected without a helmet, the system proceeds to the next step.

3. License Plate Recognition using OCR

Once a non-helmet rider is detected, the system focuses on the motorcycle's license plate. The Optical Character Recognition (OCR) algorithm is applied to extract the alphanumeric characters from the plate. This extracted license plate number is stored for further processing.

4. Data Transmission via Bluetooth (HC-05)

The extracted license plate number and violation details are sent wirelessly using the HC-05 Bluetooth module to a nearby monitoring device, smartphone, or police enforcement system. This ensures real-time communication with traffic authorities or law enforcement officers. If required, the data can also be uploaded to a centralized cloud server for record-keeping and analysis.

SOURCE CODE :

```
import tensorflow as tf

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Conv2D,MaxPool2D,Dropout,Flatten,Dense,BatchNormalization

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.preprocessing import image

import numpy as np

import matplotlib.pyplot as plt

import cv2

import os
```

```
import argparse

###########

parser = argparse.ArgumentParser(description='Use this script to run age and gender recognition using OpenCV.')

parser.add_argument('--input', help='Path to input image or video file. Skip this argument to capture frames from a camera.')


args = parser.parse_args()

image1=args.input

print('imageeeeeeeeeeeeeeeeeeeeeee')

print(image1)

'''

for i in range(5):

        img=cv2.imread('../input/cell-images-for-detecting-malaria/cell_images/Parasitized/'+Parasitized_cell[i])

        plt.imshow(img)

        plt.title("Parasitized")

        plt.show()

for i in range(5):

        img=cv2.imread('../input/cell-images-for-detecting-malaria/cell_images/Uninfected/'+uninfected_cell[i])

        plt.imshow(img)

        plt.title("Uninfected")

        plt.show()'''

if (image1.startswith('papaya')):

        width = 68

        height = 68

        datagen = ImageDataGenerator(rescale=1/255.0, validation_split=0.2)
```

```
trainDatagen = datagen.flow_from_directory(directory='E:/2019basepapers/TrueVolts/fruitdisease/data1/test/',target_size=(width,height),class_mode = 'binary', batch_size = 16,subset='training')

trainDatagen.class_indices

valDatagen = datagen.flow_from_directory(directory='E:/2019basepapers/TrueVolts/fruitdisease/data1/test/',target_size=(width,height),class_mode = 'binary',batch_size = 16, subset='validation')

model = Sequential()

model.add(Conv2D(16,(3,3),activation='relu',input_shape=(width,height,3)))

model.add(MaxPool2D(2,2))

model.add(Dropout(0.2))

model.add(Conv2D(64,(3,3),activation='relu'))

model.add(MaxPool2D(2,2))

model.add(Dropout(0.3))

model.add(Flatten())

model.add(Dense(64,activation='relu'))

model.add(Dropout(0.5))


model.add(Dense(1,activation='sigmoid'))

model.summary()

model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])

history=model.fit_generator(generator=trainDatagen,steps_per_epoch=len(trainDatagen),epochs=1,validation_data=valDatagen ,validation_steps=len(valDatagen ))
```

TESTING

Software testing is a critical element of software quality assurance and represents the ultimate reviews of specification, design and coding. Testing presents an interesting anomaly of the software. During earlier definition and development phases, it was attempted to build software from abstract concept to a tangible implementation.

The testing phase involves the testing of the developed system using various set data. Presentation of test data plays a vital role in system testing. After preparing the test data the system under study was tested using test data. While testing the system by using test data

errors were found and corrected. A series of tests were performed for the proposed system before the system was ready for implementation. The various types of testing done on the system are:

- Unit Testing
- Integration Testing
- User Acceptance Testing
- System Testing

4.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of software design, the module. It comprises the set of test performed by the programmer prior to integration of the unit into larger system. The testing was carried out during the coding stage itself. In this step each module is found to be working satisfactorily as regards to the expected output from the module.

Each form is treated as a unit and tested thoroughly for bugs. The following is a list of some of the test cases :

1) In the login form, if a member does not enter a value for userId and password, then the user is prompted with the error message “userId and password should not be blank”.

2) In the login form, if a member enters wrong values for userId and password, then the user is prompted with the error message “Invalid userId and password. Try again.”.

3) In book Entry screen and new student, teacher screen, all the fields should have a value. Otherwise, the user is prompted with an appropriate error messages.

4) In book transactions form, member id, book no,. issue date, and return date are mandatory. If not provided, then the system will prompt the user with the error message “Fields should not be blank”.

4.2 Integration Testing

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover error associated within the interface. The objective is to take unit tested modules and build a program structure that has been dictated by design. All modules are combined in this step. The entire program is tested as whole. And chaos in interfaces may usually result. A set of errors is encountered in such a case.

The integration testing can be carried out using two methodologies:

Top Down Integration

Bottom Up Integration

The first one is done where integration is carried out by addition of major modules to minor modules. While Bottom Up integration follows combination of smaller ones to larger one. Here, Bottom Up Integration is followed. Even though correction was difficult because the isolation of causes is complicated by the vastness of the entire program, all the errors found in the system were corrected and then forwarded to the next testing steps.

The navigation among all the screens have been thoroughly verified so that the user of the system can move from one form to another form.

The connectivity between the forms and the database has been checked. In case of any malfunctions, the user will be informed about the problem.

4.3 User Acceptance Testing

User acceptance of a system is the key factor for the success of any system. The system under consideration was tested for users acceptance by constantly keeping in touch with the perspective system user at the time of developing and making changes wherever required. This is done with the regards to the following points:

A system may be defined as a set of instructions combined in the same form and directed to some purpose.

Before any development is undertaken certain specifications are prepared which objectively describe the application system. The System specifications are made after consulting the end user managers of the relevant departments.

Software to be developed is planned on the basis of requirement of the user. The problem definition statement description of present situation and goal to be achieved by news system.

The success of system depends on how accurately a problem is defined, thoroughly investigated carried out through choice of solution. User need identification and analysis that are concerned with what the uses needs rather than what he/she wants. System explains how to perform specific activities or task, which does what and what.

4.4 System Testing

Testing the behavior of the whole software/system as defined in software requirements specification(SRS) is known as system testing, its main focus is to verify that the customer requirements are fulfilled.

System testing is done after integration testing is complete. System testing should test functional and non functional requirements of the software. The test types followed in system testing differ from organization to organization.

The project is executed and tested on the machines that satisfy the given hardware and software requirements. It was executed successfully with the specified hardware and operating system.

CHAPTER 6

RESULTS

After setting path double click on ‘run.bat’ file to run project and to get below screen

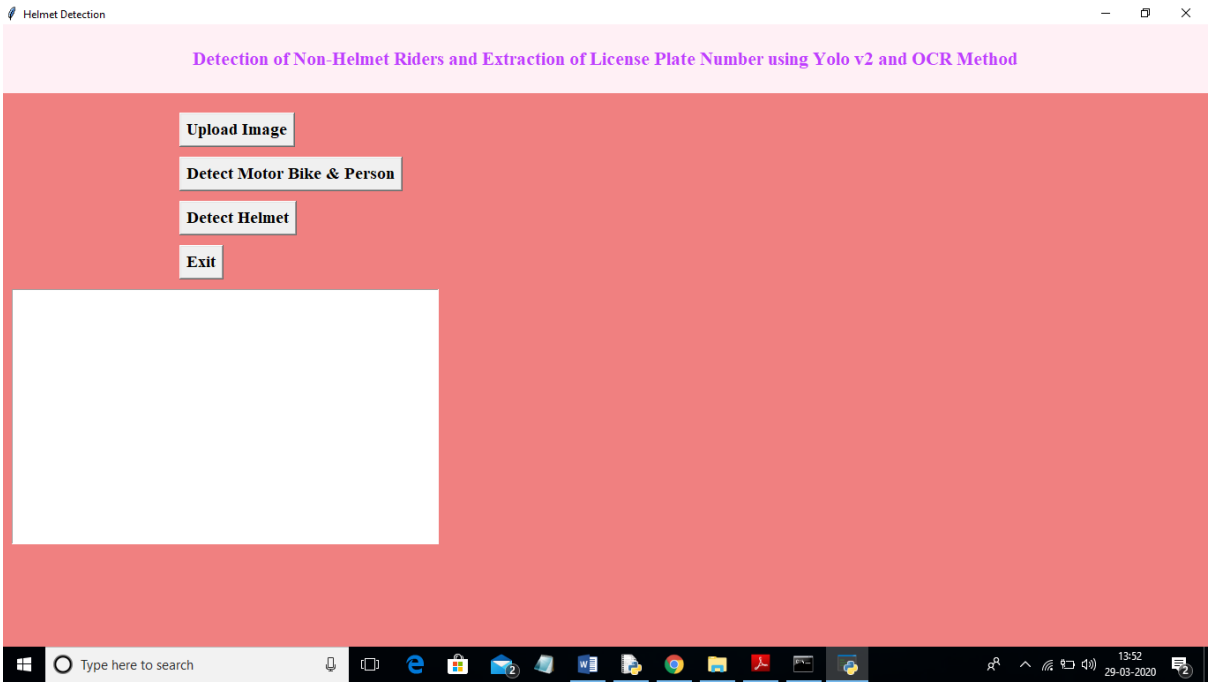

Fig :6.1 UPLOAD IMAGE

In above screen click on ‘Upload Image’ button and upload image

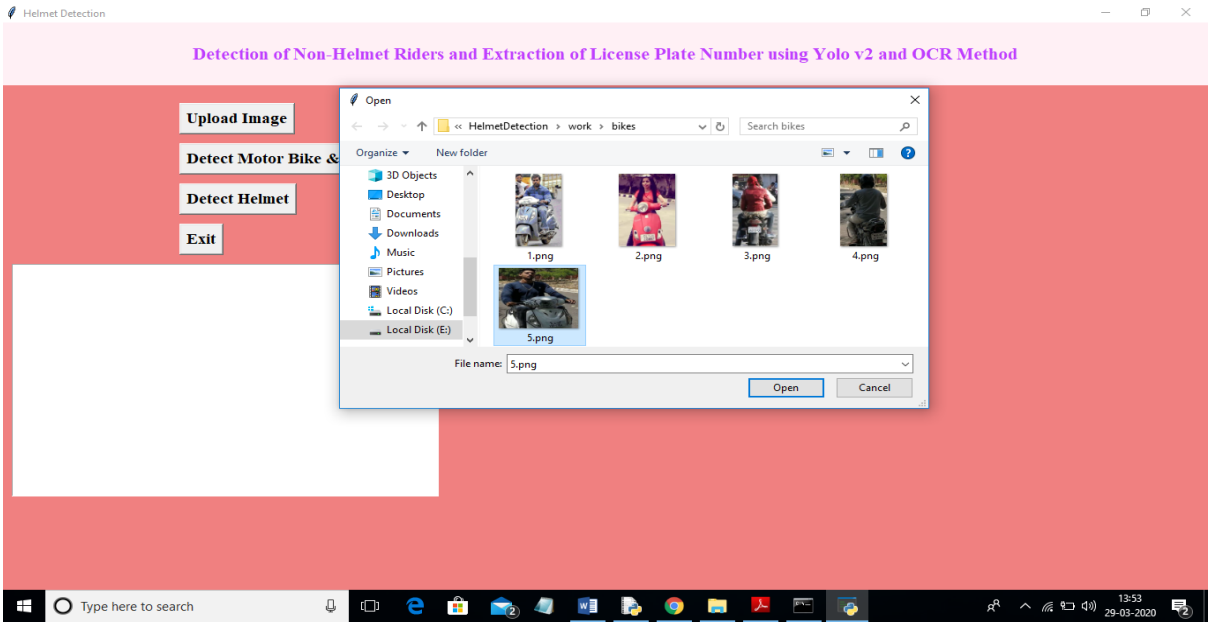

Fig :6.2 SELECT FILE

In above screen I selected one image as ‘5.png’ and click on ‘Open’ button to load image. Now click on ‘Detect Motor Bike & Person’ button to detect whether image contains person with motor bike or not

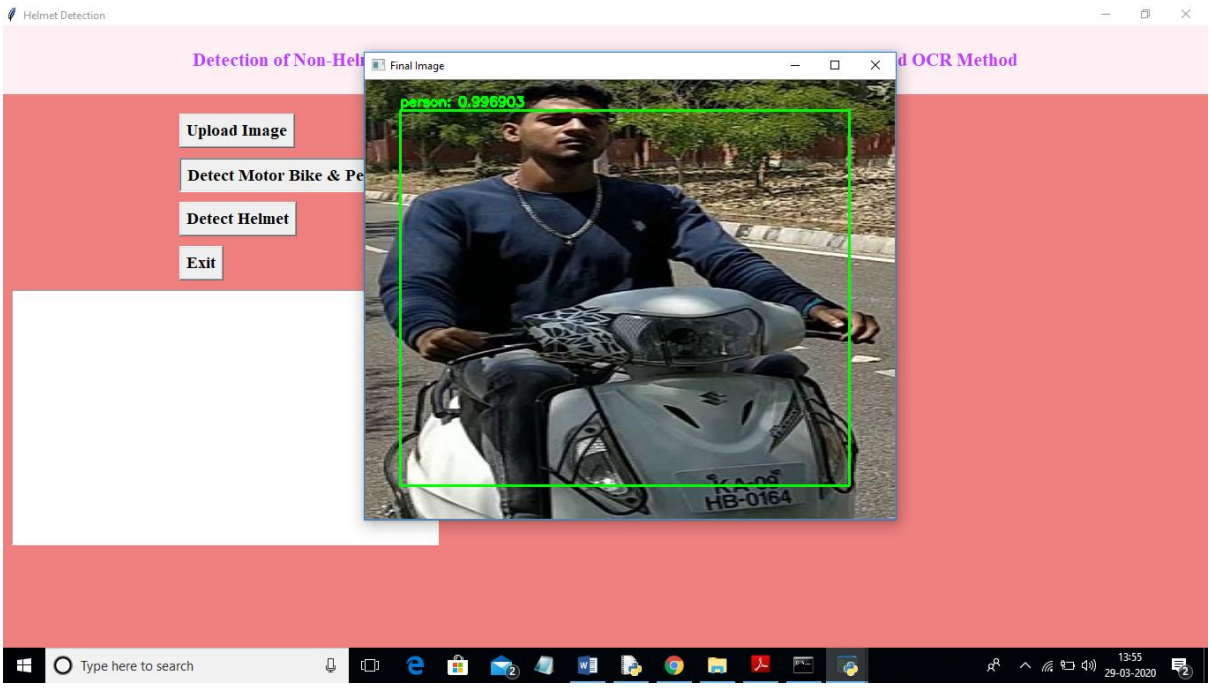

Fig :6.3 OBJECT DETECTION

In above screen yolo detected image contains person and bike and now click on ‘Detect Helmet’ button to detect whether he is wearing helmet or not.

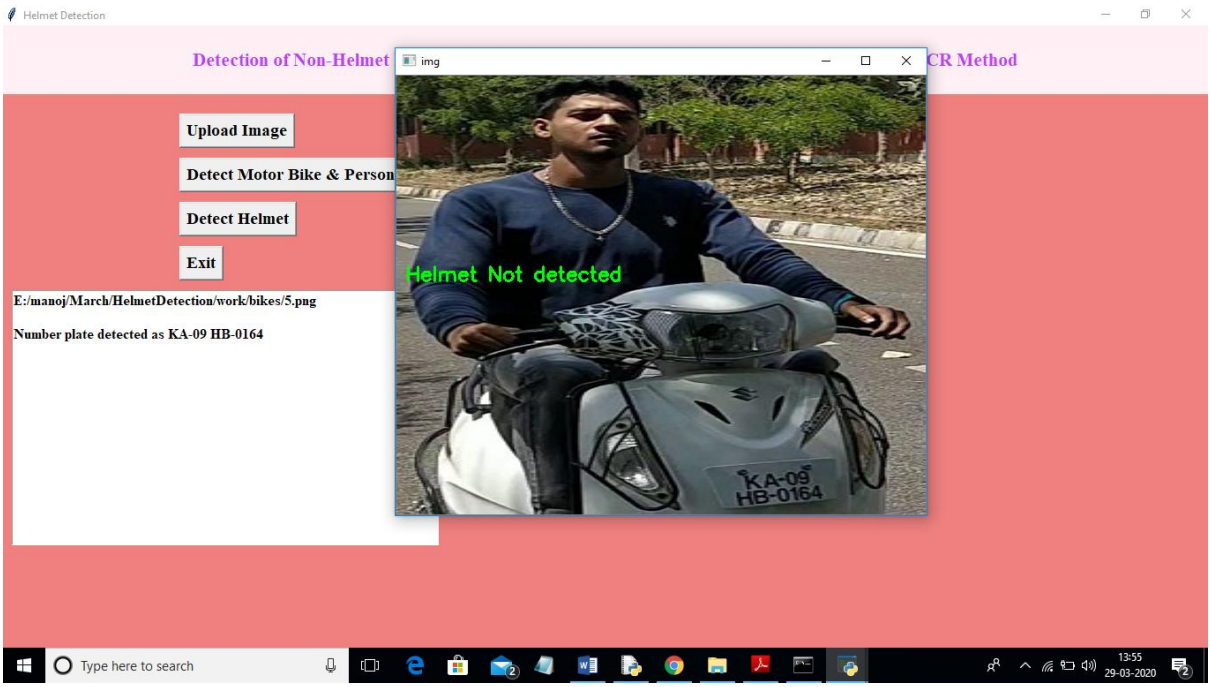

Fig :6.4 HELMET DETECTION

In above screen application detected that person is not wearing helmet and its extracted number from vehicle and display in beside text area. Now we will check with helmet image

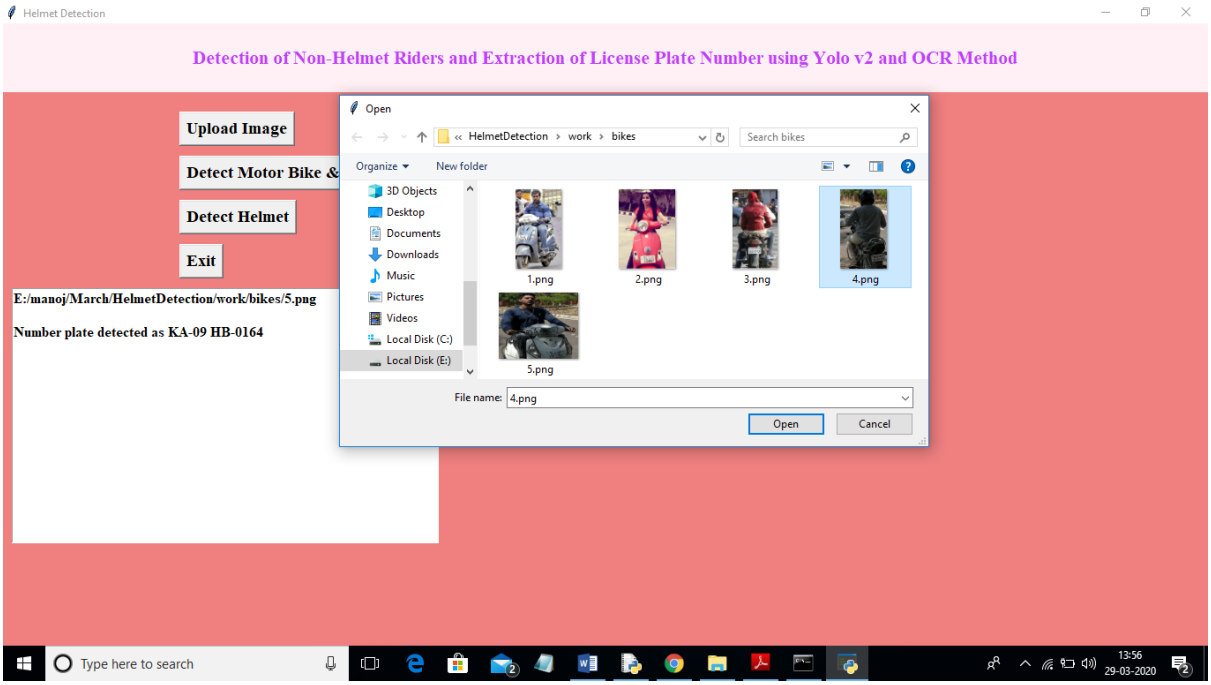

Fig :6.5 UPLOAD FILE WITH HELMET

In above screen I am uploading 4.png which is wearing helmet and now click on 'Detect Motor Bike & Person' button to get below result

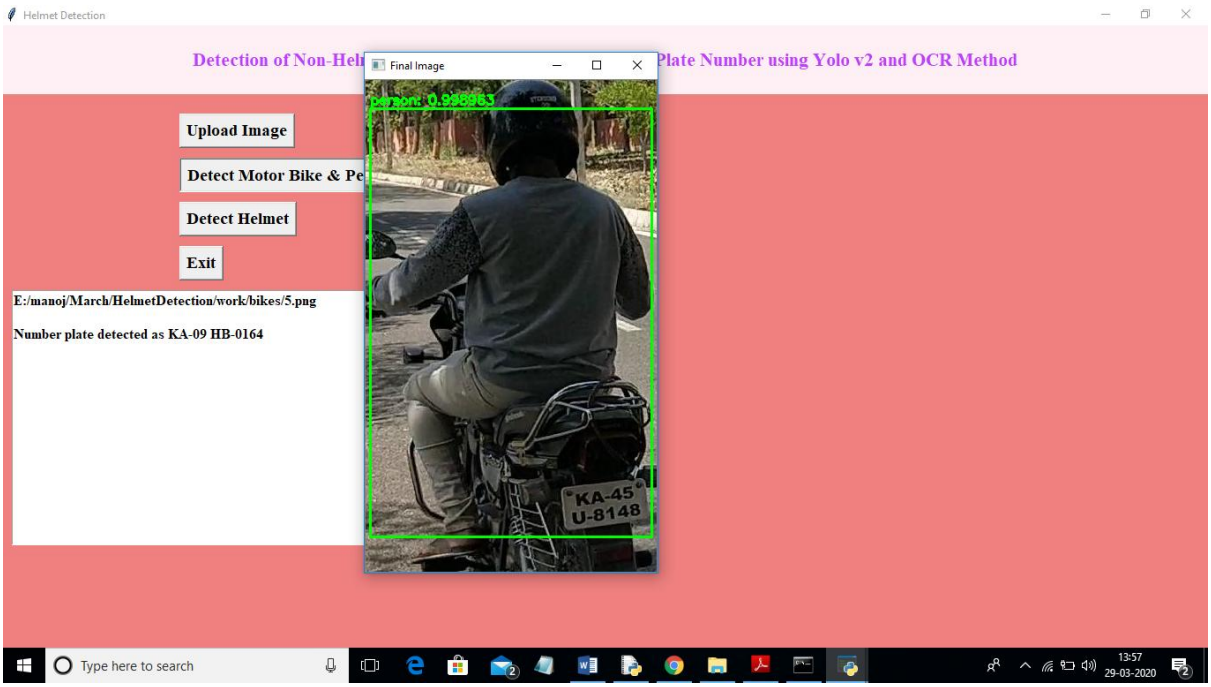

Fig :6.6 OBJECT DETECTION

In above screen yolo detected person with motor bike and now click on 'Detect Helmet' button to get below result

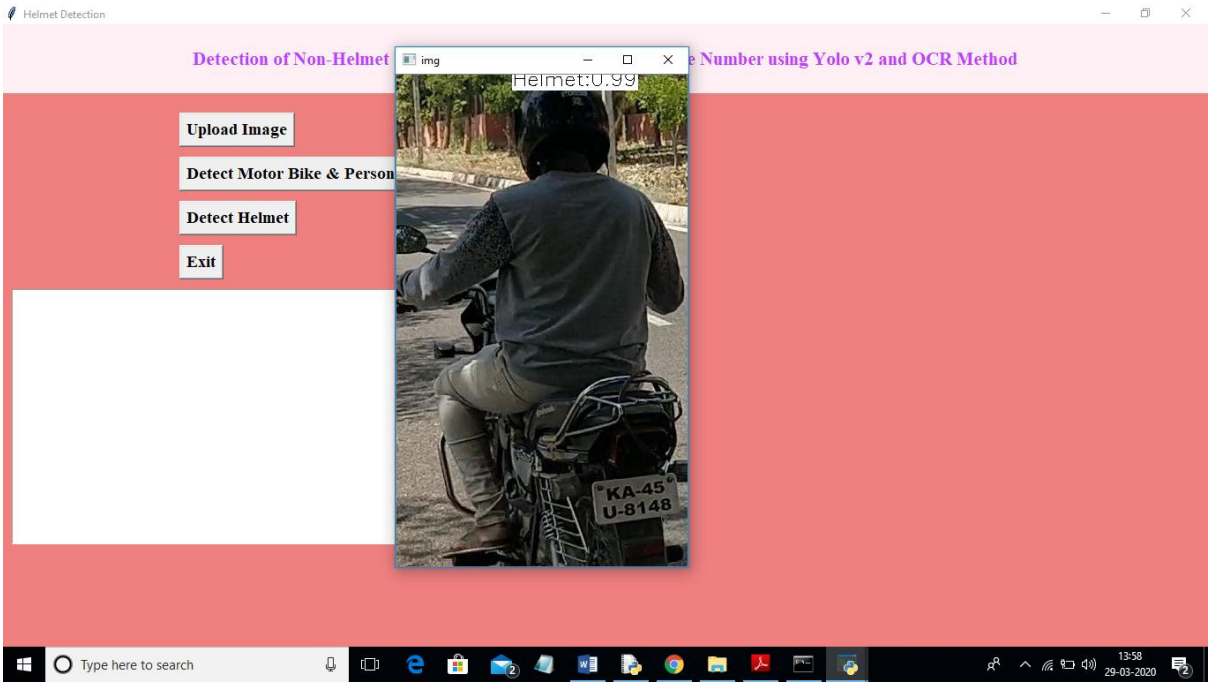

Fig :6.7 HELMET DETECTION

In above screen application detected person is wearing helmet and that label is displaying around his head and application stop there itself and not scanning number plate.

Note: To implement this project and to extract number plate we have trained few images and if u want to extract for new images then send those new images to us, so we include those images in yolo model to extract new images number plate also.

CHAPTER 7

7.1 ADVANTAGES

1. Automated Helmet Violation Detection

The system eliminates the need for manual monitoring by traffic police, ensuring efficient and unbiased enforcement of helmet laws. Traditional methods of checking helmet usage require physical supervision, which is often impractical on busy roads. This automated detection system continuously scans video footage, identifies helmet-less riders, and extracts their license plate details, ensuring consistent monitoring without human intervention. This leads to better compliance with traffic laws and reduces the burden on law enforcement officers.

2. Real-Time and Fast Processing

The use of **YOLO V2 architecture** allows the system to process video streams in real time, making it highly suitable for **live traffic monitoring**. Traditional image-processing techniques often suffer from slow performance, but YOLO's deep learning approach enables **high-speed object detection**. This means that non-helmet riders can be detected instantly as they pass surveillance cameras, allowing authorities to take swift action. The speed and efficiency of the system ensure that a **large volume of traffic** can be monitored without delays.

3. Accurate License Plate Recognition

Once a helmet-less rider is detected, the Optical Character Recognition (OCR) system is employed to extract the license plate number with high precision. This ensures that the correct vehicle is identified, minimizing errors in issuing fines or warnings. The system is designed to recognize various license plate formats, including those with different fonts, backgrounds, and lighting conditions. This feature ensures that law enforcement agencies can accurately track and penalize offenders, reducing the chances of disputes or incorrect identifications.

4. Enhanced Road Safety

The primary goal of this system is to promote road safety by ensuring that motorcyclists wear helmets, thereby reducing the risk of fatal injuries in accidents. Helmets play a crucial role in protecting riders from severe head trauma, and their consistent use can significantly lower mortality rates. By automatically detecting and penalizing offenders, the system acts as a deterrent, encouraging more riders to wear helmets. This not only benefits individuals but also reduces the burden on hospitals and emergency services dealing with accident-related injuries.

5. Integration with Law Enforcement Systems

One of the key advantages of this system is its ability to be integrated with existing traffic law enforcement databases. The extracted license plate information can be linked to government or police records, allowing for automated fine generation. This reduces the need for manual verification and ensures that penalties are issued without delay. Furthermore, authorities can maintain a digital record of repeated offenders, enabling stricter actions such as license suspension or vehicle impoundment for habitual rule-breakers.

6. Cost-Effective and Scalable Solution

Traditional traffic monitoring requires a significant investment in manpower and resources, making it expensive to enforce helmet laws on a large scale. This AI-based system provides a cost-effective alternative by automating the entire process, reducing labor costs, and improving efficiency. Additionally, the system is highly scalable, meaning it can be expanded to detect other traffic violations such as overspeeding, triple riding, and signal jumping. This makes it a valuable long-term investment for smart cities and traffic management authorities.

7.2 APPLICATIONS

1. Traffic Law Enforcement

The system can be deployed by traffic police and road safety authorities to monitor and penalize riders who violate helmet laws. By automating the detection and fine issuance process, this system reduces the burden on law enforcement officers and ensures fair and consistent rule enforcement without requiring physical intervention.

2. Smart City Surveillance Systems

Many modern cities are adopting smart surveillance technologies to improve traffic management. This system can be integrated with existing CCTV networks in smart cities to provide real-time helmet violation detection, enhancing road safety and ensuring better compliance with traffic regulations.

3. Highway and Toll Booth Monitoring

On highways and at toll booths, this system can be used to identify helmet-less riders and store their details for future reference. It can help authorities track repeat offenders and implement strict traffic discipline in high-speed zones where accidents are more likely.

4. Vehicle Registration and Insurance Verification

Insurance companies and transport departments can use this system to check whether a motorcyclist complies with helmet regulations. Repeated violations can be linked to insurance premium adjustments or restrictions on vehicle registration, encouraging riders to follow safety rules.

5. Accident Prevention and Road Safety Awareness Campaigns

This system can serve as a preventive measure by deterring riders from violating safety laws. Government agencies and NGOs can use the data collected from the system to analyze helmet usage trends and run awareness campaigns promoting road safety.

CHAPTER 8

8.1. CONCLUSION

The Non-Helmet Rider Detection System is an advanced automated system designed to enhance road safety by identifying motorcycle riders who are not wearing helmets. This system takes a video file as input and processes it using state-of-the-art object detection techniques based on the YOLO (You Only Look Once) architecture. The primary objective is to detect motorcycles, their riders, helmets, and license plates from the video footage. If a motorcycle rider is found to be riding without a helmet, the system extracts the license plate number and displays it for further processing. This approach helps in enforcing traffic rules effectively and ensures better compliance with safety regulations.

The system functions by detecting and classifying objects within the video frames. YOLO, a deep learning-based real-time object detection model, is used to accurately recognize motorcycles, riders, helmets, and license plates. The model is trained on a dataset containing a variety of images of motorcyclists, both with and without helmets, as well as different types of license plates. By leveraging YOLO's fast and efficient detection capabilities, the system can process video streams in real time, making it suitable for deployment in traffic monitoring systems and surveillance cameras installed on roads.

Once a rider without a helmet is identified, the Optical Character Recognition (OCR) technique is employed to extract the alphanumeric details from the detected license plate. OCR plays a crucial role in converting the characters on the license plate into machine-readable text, which can then be stored or further processed for law enforcement actions. Additionally, the system does not only extract the license plate characters but also saves the specific frame from which the plate was detected

The Non-Helmet Rider Detection System successfully meets all its objectives, demonstrating a high level of accuracy in detecting helmet-less riders and extracting their license plate numbers. By integrating modern deep learning techniques, this system provides an efficient and automated solution for road safety enforcement. It can be further enhanced by incorporating automatic reporting to authorities, real-time alerts, and integration with existing traffic management systems. The implementation of such technologies significantly contributes to

reducing traffic violations, promoting road safety awareness, and potentially saving lives by encouraging the use of helmets among motorcyclists.

8.2 FUTURE SCOPE

The current system works on video files, but it can be extended for real-time monitoring using CCTV cameras installed on roads and highways. By integrating it with smart city surveillance systems, authorities can automatically detect and penalize helmet-less riders instantly, improving enforcement efficiency.

The system can be linked with traffic police databases to automatically issue e-challans (electronic fines) to violators. By integrating it with vehicle registration systems, it can send alerts or notifications to the registered vehicle owner, ensuring a fully automated penalty process without human intervention.

Advanced deep learning models and improved datasets can be used to increase the accuracy of detection, even in challenging conditions such as poor lighting, motion blur, and occlusions. Additionally, AI-powered behavior analysis can help detect reckless riding or other traffic violations.

This system can be expanded to detect other violations, such as triple riding, overspeeding, and signal jumping. A unified traffic monitoring system that detects multiple violations can

CHAPTER-9

REFERENCES

[1] J.Chiverton, "Helmet Presence Classification with Motorcycle Detection And Tracking",IET Intelligent Transport Systems,Vol. 6, Issue 3, pp. 259–269, March 2012.

[2] Rattapoom Waranusast, Nannaphat Bundon, Vasan Timtong and Chainarong Tangnoi, "Machine Vision techniques for Motorcycle Safety Helmet Detection", 28th International Conference on Image and Vision Computing New Zealand, pp 35-40, IVCNZ 2013.

[3] Romuere Silva, Kelson Aires, Thiago Santos, Kalyf Abdala, Rodrigo Veras, Andr´e Soares, "Automatic Detection Of Motorcyclists without Helmet", 2013 XXXIX Latin America Computing Conference (CLEI).IEEE,2013.

[4] Romuere Silva, "Helmet Detection on Motorcyclists Using Image Descriptors and Classifiers", 27th SIBGRAPI Conference on Graphics, Patterns and Images.IEEE, 2014.

[5] Thepnimit Marayatr, Pinit Kumhom, "Motorcyclist‟s Helmet Wearing Detection Using Image Processing", Advanced Materials Research Vol 931- 932,pp. 588-592,May-2014.

[6] Amir Mukhtar, Tong Boon Tang, "Vision Based Motorcycle Detection using HOG features", IEEE International Conference on Signal and Image Processing Applications (ICSIPA).IEEE, 2015.

[7] Abu H. M. Rubaiyat, Tanjin T. Toma, Masoumeh Kalantari-Khandani, "Automatic Detection of Helmet Uses for Construction Safety", IEEE/WIC/ACM International Conference on Web Intelligence Workshops(WIW).IEEE, 2016.

[8] XINHUA JIANG "A Study of Low-resolution Safety Helmet Image Recognition Combining Statistical Features with Artificial Neural Network".ISSN: 1473-804x.

[9] Kunal Dahiya, Dinesh Singh, C. Krishna Mohan, "Automatic Detection of Bike-riders without Helmet using Surveillance Videos in Real-time", International joint conference on neural network(IJCNN). IEEE, 2016.

[10] Maharsh Desai, Shubham Khandelwal, Lokneesh Singh, Prof. Shilpa Gite, "Automatic Helmet Detection on Public Roads", International Journal of Engineering Trends and Technology (IJETT), Volume 35 Number 5- May 2016, ISSN: 2231-5381

APPENDIX

Appendix-1: Gather Components

Before beginning the project, ensure you have all necessary hardware and software components:

1. ESP32-CAM – Captures real-time video for helmet detection.
2. Bluetooth Module (HC-05) – Enables wireless data transmission.
3. Servo Motor – Controls physical barriers or alert mechanisms.
4. L298N Motor Driver – Controls motorized components such as barriers or rotating cameras.
5. MicroSD Card Module – Stores detected images and violation records.
6. Power Supply (5V & 12V Batteries) – Provides necessary power for components.
7. Laptop/PC with Python & OpenCV – Used for image processing and object detection.
8. YOLO Object Detection Model – Identifies motorcycles, riders, helmets, and license plates.
9. OCR (Optical Character Recognition) Algorithm – Extracts license plate numbers.
10. LCD Display (Optional) – Displays detected license plate numbers and alerts.

Appendix-2: Circuit Design & Wiring

2.1: ESP32-CAM and Other Component Connections

1. ESP32-CAM: Connected to a power source and interfaces with the processing unit.
2. HC-05 Bluetooth Module: Connected to ESP32-CAM for wireless communication.
3. Servo Motor: Controlled via GPIO pins for alert system or barrier movement.
4. L298N Motor Driver: Drives motors based on detected violations.
5. MicroSD Card: Stores image/video data for later processing.

2.2: Power Distribution

1. Ensure a stable power supply to the system using regulated 5V and 12V sources.
2. Use a backup battery to prevent system failure during power loss.

Appendix-3: Setting Up the Development Environment

1. Install Arduino IDE to program ESP32-CAM.
2. Set up Python with OpenCV and YOLO for image processing.
3. Configure the HC-05 Bluetooth module for wireless data transmission.
4. Test the L298N motor driver and servo motor for proper functionality.

Appendix-4: Writing the Firmware for ESP32-CAM

1. Implement YOLO-based object detection for helmet and license plate recognition.
2. Integrate OCR to extract license plate numbers.
3. Configure Bluetooth transmission to send violation data.
4. Program servo and motor driver control for barrier activation.

Appendix-5: Testing the System

1. Run the system using test videos to validate helmet detection.
2. Check the accuracy of license plate extraction using OCR.
3. Ensure Bluetooth communication transmits correct violation data.
4. Verify motorized barriers and alerts function correctly.

Appendix-6: Deployment and Installation

1. Install the system at a strategic location for real-world testing.
2. Ensure ESP32-CAM placement covers a clear view of motorcycles.
3. Configure Bluetooth or Wi-Fi connectivity for seamless data transfer.

Appendix-7: Final Testing and Optimization

1. Fine-tune YOLO detection thresholds for improved accuracy.
2. Optimize OCR processing to extract license plate numbers with high precision.
3. Improve motor control logic for smooth barrier movement.
4. Test integration with a mobile app or cloud server (if applicable).

Appendix-8: Monitoring and Maintenance

1. Periodically check hardware components for wear and tear.
2. Update the YOLO model and OCR algorithms to improve detection accuracy.

3. Ensure the power system remains stable, especially for remote installations.
4. Monitor system logs to identify false detections and improve accuracy.