Smart Traffic Management System

By

Yusuf Abdullah
22321077
Zayed Al Hossain Fahim
21221059
Ehsanul Munir Rodro
18121113
Fahmidul Hassan Abir
19121013

A Final Year Design Project (FYDP) submitted to the Department of Electrical and Electronic Engineering in partial fulfillment of the requirements for the degree of BSEEE

Electrical and Electronic Engineering,
BRAC University
January, 2026

Smart Traffic Management System

By

Yusuf Abdullah
22321077
Zayed Al Hossain Fahim
21221059
Ehsanul Munir Rodro
18121113
Fahmidul Hassan Abir
19121013

A Final Year Design Project (FYDP) submitted to the Department of Electrical and Electronic Engineering in partial fulfillment of the requirements for the degree of BSEEE

Academic Technical Committee (ATC) Panel Member:

Dr. Md Golam Sorwar Hossain (Chair)

Professor, Department of EEE, BRAC University

Md. Saif Kabir (Member)

Senior Lecturer, Department of EEE, BRAC University

Junaid Jalal (Member)

Lecturer, Department of EEE, BRAC University

Electrical and Electronic Engineering,
BRAC University
January, 2026

Declaration

It is hereby declared that

1. The Final Year Design Project (FYDP) submitted is my/our own original work while completing degree at Brac University.
2. The Final Year Design Project (FYDP) does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The Final Year Design Project (FYDP) does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Yusuf Abdullah ID: 22321077	Zayed Al Hossain Fahim ID: 21221059
Ehsanul Munir Rodro ID: 18121113	Fahmidul Hassan Abir ID: 19121013

Approval

The Final Year Design Project (FYDP) titled “Smart Traffic Management System” submitted by

1. Yusuf Abdullah (22321077)
2. Zayed Al Hossain Fahim (21221059)
3. Ehsanul Munir Rodro (18121113)
4. Fahmidul Hassan Abir (19121013)

of Fall, 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of BSEEE on 05th January, 2026

Examining Committee:

Academic Technical Committee (ATC):
(Chair)

_____
Dr. Md Golam Sorwar Hossain
Professor,
Department of EEE, BRAC University

Final Year Design Project Coordination Committee:
(Chair)

_____
Dr. Mirza Rasheduzzaman
Associate Professor
Department of EEE, BRAC University

Department Chair:

_____
Dr. Md. Mosaddequr Rahman
Chairperson
Department of Electrical and Electronic Engineering

Ethics Statement

This report submitted to Turnitin achieved a similarity score of only 10%, mainly due to standard academic contents such as chapter headings, references, tables, and technical specifications. The AI-generated content score was also verified to be below 20%.

Abstract

This project aims to design and develop a smart traffic management system for a four-way intersection to improve overall traffic efficiency, emergency response, and road safety. The suggested system dynamically manages traffic lights based on the number of vehicles in each lane and only allows a green signal to the lane with the most vehicles. It contains a camera-based approach using a pre-trained YOLO model to detect vehicle congestion, classify emergency vehicles, and give signal priority even if no vehicle is present in the other lanes. To increase the reliability of the system, a hybrid backup architecture is used. If the camera-based detection fails, the microcontroller-based IIoT sensor network is automatically activated, and congestion and emergency vehicle detection continue. Furthermore, a special microcontroller continuously monitors wrong-way vehicle movement and raises an alarm when a wrong-way vehicle is detected, thereby enhancing traffic safety. The proposed system features AI, embedded systems, and multiple sensors as redundant features, making it robust, scalable, and compatible with modern intelligent transportation systems.

Keywords: Intelligent Transportation System (ITS), Deep Learning, YOLOv8, Real-Time Traffic Monitoring, Adaptive Signal Control, Embedded AI, Sensor Fusion, Fault-Tolerant Architecture, Smart City Traffic Management

Dedication

This project is dedicated to our parents, teachers and well wishers who have inspired us throughout and helped us to grow, through constant support, encouragement and sacrifice.

Finally, this work is dedicated to all researchers, engineers and innovators who contribute to the development of intelligent and safer transportation systems to make smarter and more sustainable cities.

Lastly, we honor the future of technology and engineering innovations that enhance public safety, efficient urban mobility and improve society.

Acknowledgement

We would like to thank our ATC panel members and our esteemed faculty members for providing us with constant encouragement, guidance and useful tips throughout this project. We are also grateful to the Department of Electrical and Electronic Engineering (EEE), BRAC University, for providing us with the opportunity and resources to successfully complete this work.

Table Of Contents

Chapter 1 : Introduction

1.1 Introduction:

The intersections are significant control points in urban transport networks, particularly for the 4-way traffic intersections, in which traffic flow with multiple directions needs to be controlled simultaneously. The growing number of vehicles and the increasing urban mobility requirements have made urban traffic management (UTM) increasingly important for the efficient and safe use of roads, traffic flow and road safety. So, transportation systems are step by step shifting from the old-fashioned time-based traffic control to smart and adaptive traffic management solutions.

With the latest advances in computer vision, embedded systems and Artificial Intelligence, real-time monitoring and decision-making for traffic have become more feasible than ever before. The vision-based system with cameras and deep learning models allows for precise vehicle detection and classification, density estimation, and special vehicle identification, like fire trucks and ambulances. Technologies allow traffic signals to change the length of time they are on based on the real time traffic conditions, rather than set time cycles.

Meanwhile, embedded microcontroller systems and sensor networks based on the Internet of Things (IoT) including the infrared and ultrasonic sensors are shown to be effective for detecting the presence of vehicles at a localized level, monitoring lanes, and checking their safety level. These sensor based systems are perfect for real-time alerts, low power consumption and continuous operation. They are essential to the system reliability and together with the advanced vision based systems, to the system robustness.

The prioritization of emergency vehicles and enforcement of traffic laws are other important aspects of intelligent transportation systems. Automatic detection of emergency vehicles can help get intersections cleared more quickly and real-time monitoring of the wrong way vehicle movement can help improve traffic safety by alerting and preventing dangerous situations as they happen. The fusion of these aspects into a holistic traffic management system can help enhance urban traffic safety and efficiency.

This paper proposes the design and development of a smart traffic management system which involves vision based vehicle analysis and sensor modules based on microcontroller at a four way intersection. The proposed system highlights adaptive signal control, right-of-way user prioritization, wrong way direction detection and system reliability, using a hybrid architecture. The goal of the work is to show a scalable and viable prototype that can be used for future development of intelligent traffic infrastructure.

1.2 Problem Statement:

Most current fixed-time/semi-actuated traffic signal systems at four-way intersections are not responsive to the changing traffic volume or traffic conditions in real-time. These systems, however, are static and are unable to adjust to the changing vehicle densities between lanes dynamically and therefore are not able to allocate signals efficiently, resulting in delays to the

vehicle queuing and inefficient use of road space. Lack of real-time congestion based decision making is a key constraint to existing intersection traffic management infrastructures.

Furthermore, traditional traffic control systems cannot provide an automatic and reliable solution for detecting and prioritizing emergency vehicles. Traffic lights are often unable to recognize the presence of emergency vehicles or adjust their normal operating mode, forcing emergency vehicles to stop at signals. This limitation is especially harmful during emergencies, where response time is critical, creating a significant operational gap in current traffic management systems.

One of the current problems that are not solved in the current intersection control is the inability to automatically detect when vehicles are driving in the wrong direction. Wrong-way driving can be a significant safety issue and cause serious crashes and traffic disruptions. Existing systems primarily use human intervention and enforcement post event and there is no inherent real-time detection and alerting system to minimize the risk as it is occurring.

Furthermore, vision-based intelligent traffic systems that are only based on camera input and machine learning algorithms are also prone to failure, for example, due to low visibility, weather conditions, failure of the hardware or communication. Without fault-tolerant back-up system, the system is not reliable and cannot be deployed in real world.

So this project aims to fill the gap between current traffic management solutions, which fail to meet these goals, and achieve the ultimate goal of a reliable, real-time, fault-tolerant traffic management system for four-way intersections that can implement adaptive congestion-based signal control, automatic emergency vehicle prioritization, real-time wrong-way movement detection, and continued operation in the event of sensor or vision system failure. The solution to the problem requires an integrated hybrid architecture of vision based intelligence and microcontroller based sensor systems to offer a sense of accuracy, safety and operational robustness.

1.3 Background study:

Traffic signal control at road intersections is one of the basic areas of Intelligent Transportation Systems (ITS). Conventional traffic control techniques were initially designed using fixed time signal plans, which were based on historical traffic data and mathematical traffic models. The delay model of Webster was used as the basis for the early design of signal timings and it is still cited in classical traffic engineering literature [1]. Although simple and predictable, these methods cannot function in presence of non-uniform and time varying traffic pattern which exists in the current urban scenario.

Actuated/adaptive traffic signal control introduced to improve responsiveness. Real-time vehicle detection technologies such as inductive loop detectors, infrared (IR) sensors, magnetic sensors and ultrasonic sensors are used for these systems. Inductive loops are intrusive and require maintenance so have been widely used. The non-intrusive techniques like IR and ultrasonic sensors are also widely used in small scale prototypes and embedded systems but they will only give limited information about the vehicle, such as whether it is present or an approximate distance, but not classify the vehicle.

With the advent of digital imaging and computing, vision-based traffic monitoring has become a trend in research. The camera-based systems can enable the complete analysis of the traffic scenario, including detection and tracking of vehicles, estimation of traffic congestion level in the lanes and its counting. Early vision based methods were background subtraction and optical flow methods [4]. In recent times, deep learning approaches with Convolutional Neural Networks (CNNs) have achieved excellent levels of accuracy and robustness on detection tasks.

In the field of deep learning based object detection frameworks, the You Only Look Once (YOLO) family of algorithms have emerged as the de facto standard for real-time applications in the field of traffic. For vehicle detection and vehicle classification in real-time at an intersection level YOLO based systems have demonstrated excellent results [5]. Such models are often used in edge computing devices like the NVIDIA Jetson or the Raspberry Pi in practical ITS applications.

There have been several different technological solutions to emergency vehicle detection and prioritization. Some of the infrastructure based approaches are like RFID tagging, dedicated short-range communication (DSRC), vehicle-to-infrastructure (V2I) communication based on GPS information [6]. Although these techniques are effective, they rely on the configuration of particular on-board equipment and on deployment of infrastructure. But, vision-based methods rely on the visual characteristics such as flashing light, shape and colour pattern of the vehicle to detect emergency vehicles, which is non-intrusive and cost effective [7].

There has been a significant amount of discussion on the possibility of enhancing traffic safety via wrong-way vehicle movement detection. To identify abnormal vehicle behavior, research has been conducted in the field of trajectory-based video analysis, motion vector estimation and direction sensor fusion [8]. The systems are not generally engineered to be a component of an integrated signal control system, but are generally employed for real-time alert and warning to drivers, and traffic enforcement with a standalone subsystem approach.

Some of the important issues mentioned in recent studies to intelligent traffic systems are reliability and fault-tolerant behaviour of the system. Vision-based solutions are very powerful, but can be impacted by fog, rain, low light and camera failure. To solve such drawbacks, hybrid architectures which integrate vision-based systems and intelligence with sensor-based embedded systems have been proposed [9]. In these systems, where real-time control is an integral part,

interfacing to sensors and safety-critical functions is of critical importance, the deterministic timing behaviour and robustness of the microcontroller platforms is of critical importance.

In conclusion, the future of adaptive intersection management incorporating AI techniques alongside embedded systems and sensor fusion has demonstrated a lot of potential. Such advances pave the way for contemporary smart traffic management systems and their assessment.

1.4 Literature Gap:

Previous studies in the field of traffic management systems have achieved great progress on enhancing the efficiency of urban intersections. Congestion management can be done by simple fixed-time traffic signal control and actuated traffic signal control methods, where the vision-based method can be used to detect and classify the vehicles in real-time and machine learning models like YOLO can be used. Likewise, systems based on sensors, which rely on infrared and ultrasonic technology are practical and low cost solutions for vehicle presence detection. Emergency vehicles and wrong way vehicles have also been investigated as priority separately using different vision based algorithms and methods of V2I communication and trajectory analysis.

But, there are certain drawbacks to these developments. Firstly, most intelligent traffic systems are built around one capability, for instance, congestion management, emergency vehicle prioritisation, wrong way vehicle detection, but not in one system. There are currently no available solutions that offer a full traffic control solution at the intersection level. Secondly, vision-based techniques are a good solution for accuracy, but they are also influenced by other factors such as lighting, weather, and even hardware failures. The few studies that include a hybrid architecture which includes sensor-based microcontroller modules and vision-based systems for providing fault tolerance are rare. Thirdly, the most effective emergency vehicle detection systems depend on specialized onboard infrastructure (RFID tags, V2I communication) or only use visual features, which are not feasible and reliable in real-world urban settings. Finally, there is limited research on real-time lane-prioritized adaptive signal control (APC) coupled with the wrong-way detection and emergency vehicle handling system for a multi-directional traffic flow environment, especially at four-way intersections.

Therefore, a smart traffic management system with hybrid, comprehensive and fault tolerant features capable of performing real time congestion based signal allocation, emergency vehicle priority, wrong way vehicle detection, and fault tolerant operations in the event of sensor and vision system failures has not been reported in the literature. Filling this gap is the basis for the current project, and it offers a chance for practical, scalable and intelligent traffic solutions that could be used in urban intersections.

1.5 Relevance to Current and Future Commercial Platforms:

To manage traffic efficiently is one of the essential components of an updated urban infrastructure, directly impacting the flow of vehicles, the safety of roads, the environment and the productivity of the economy. With increasingly high urban population grows the traffic congestion at crossings becomes more and more important, especially when it demands intelligent operating solutions with the ability to adapt in real-time. Commercial systems such as SCATS (Sydney Coordinated Adaptive Traffic System), SCOOT (Split Cycle Offset Optimization Technique) and Siemens' systems for adaptive traffic control are proven to be industry benchmarks for optimizing traffic at a large scale. In general, need sensor networks, central signal timing processor and pre-programmable adaptive algorithms that optimize signal timing using information from the vehicles. These platforms are very effective in some metros, but are often expensive to install and maintain, need extensive calibration work and depend on proprietary hardware and software infrastructures [1,2]. Not to mention, many commercially available systems are inherently designed to be at the macro-scale deployment and may not really solve the localized intersection problem especially for smaller intersections and intersections that are very dynamic.

This adds an additional focus on the application of machine learning, computer vision and IoT based sensors, which can be used in addition to the existing approaches to the emerging intelligent transportation systems. Camera-based vehicle detection and classification enables a congestion analysis, identification of vehicle types and even emergency vehicle recognition on the lanes in real-time using deep learning algorithms, such as the YOLO (You Only Look Once) algorithm [3]. Simultaneously, the embedded system incorporating the infrared, ultrasonic or magnetic sensor, which is based on a microcontroller, can provide low-cost real-time monitoring and control of traffic signals, often with a quick deployment time. What is missing in the literature and in current commercial applications, however, is typically focused on the individual function, such as congestion control or enforcing safety rules or giving preferential access for emergency vehicles. There are very few systems that bring all these elements together and can help solve traffic efficiency, emergency response, and safety compliance at the intersection level, while also tackling these issues collectively in a unified and fault-tolerant platform.

The proposed functional set of traffic control technologies here will be relevant to the current state of the art commercial products and the future smart cities needs. It comprises two components: a vision-based AI solution, which includes a sensor module, for real-time traffic congestion analysis, emergency vehicle detection and identification of cars traveling the wrong way; and an intelligent signal control solution. Unlike conventional commercial systems, it is a complete solution for four-way traffic lights for very dynamic and multidirectional traffic. Moreover, the hybrid architecture enables system fault-tolerance, operation in case of failure of the camera and sensor which would be unachievable in any commercial platform. This capability will help make this solution more useable, since it will be less likely to fail when deployed in

such scenarios as fog, rain or other conditions that could impact purely vision-based systems and may be present in a sensor failure scenario [4][5].

The system is a multi-beneficial approach from commercial and industrial aspects. It is a modular solution that can be phased-in into the existing traffic infrastructure and not require a completely new traffic infrastructure to be built, which makes it more cost efficient for the municipalities and private traffic infrastructure builders to add the modules to their existing traffic infrastructure. Use of edge platforms such as Raspberry Pi and Arduino-based microcontrollers which are readily available makes for easy adjustability, maintainability and accessibility. Moreover, the addition of prioritisation for emergency vehicles is a relevant feature for a smart city traffic management system, in the sense that there is a need to reduce the response time for emergency vehicles such as ambulances, fire trucks and police cars. The capability of wrong way driving detection is an extra measure of safety in traffic operations, and is directly applicable to commercial conformance and regulatory implementation in urban road networks.

The system is aligned with the developments of data driven, predictive and autonomous traffic management in the future commercial platforms. Future smart city deployments are expected to have machine learning-based traffic forecasting, predictive traffic congestion mitigation, autonomous management of traffic lights, and vehicle-to-infrastructure (V2I) communications. It is a smart and traditional project that can be a prototype to move forward with larger-scale smart city networks, cloud-based analytical tools and autonomous cars. The deployment aspect of the platform in itself was another area of interest as was the ease of integration it could bring to other systems.

In addition, the system is essential for industrial applications in terms of safety, reliability and cost. Self-calibration, dependency on a proprietary vendor in the use of the sensors aside, are based on open source hardware and software, meaning that there is no need for a high degree of customization from a vendor. The system is designed to be redundant, meaning it can operate even if parts of it are damaged or disrupted, thus maintaining the system's functionality and reliability even when large sections are impacted. The combination of multi-modal detection and emergency response prioritization and adaptive signaling in one, commercially available solution, not only provides a means to address the barriers in the current traffic management technology, but also provides for the needs of future urban mobility systems.

Finally, the proposed system here is applicable to the existing off-the-shelf systems and it can be also a scalable, resilient and intelligent system for future urban traffic management. It seamlessly integrates vision-based AI models, embedded sensing capabilities, and hybrid fault-tolerant design, making the leap from academia to commerce. These features are in line with the international Smart City projects, result in concrete gains in traffic, crisis response time and

safety and make it a potential candidate for incorporation into future commercial traffic control systems.

1.6 Objectives, Specifications, Requirements, and Constraints:

1.6.1 Objectives:

- Real-time monitoring and counting of vehicles at stop lights by each of the lanes.
- Optimize traffic flow by dynamically assigning green signals to the most congested lane.
- Ensure that green is given to Emergency Vehicles (Ambulance, Fire Vehicle, Police Vehicle) without considering the other lanes congestion when there is an emergency signal displayed.
- Recognize incorrect vehicle navigating in any lane and send live alerts for improved road safety.
- Extend system reliability by a hybrid solution that is vision-based AI combined with sensor-based module and microcontroller.
- Create a solution model which can be used at real urban intersections, is scalable and is modular.

1.6.2 Specifications:

Design Approach - 1 (Image Processing based management):

- **Vehicle detection:** USB camera with a pre-trained YOLO AI model that provides real-time vehicle and emergency vehicle detection.
- **Processing platform:** Raspberry Pi - AI based processing and decision-making.
- **Signal control:** Traffic lights controlled by the microcontroller with raspberry pi for adaptive green signal allocation.
- **Emergency vehicle handling:** Camera and YOLO AI identifies emergency vehicles and automatically switches traffic signals for immediate priority.
- **Wrong way movement detection:** Standby Arduino Uno microcontroller continuously scans the lanes for wrong way vehicles and initiates alerts.
- **Response time:** detection to adjustment to the signal in 1-2 seconds for normal vehicles and less than 1 second for emergency vehicles.
- **Power supply:** 5V-12V DC for Raspberry Pi, Arduino, sensors and traffic lights.
- **Environmental requirements:** Normal visibility conditions, (performance may be affected under low light or adverse weather conditions).

Design Approach 2 (Hybrid Vision-Sensor Backup System):

- **Emergency vehicle detection backup:** Backup detection of priority vehicles with predefined vehicle parameters (e.g., vehicle size, vehicle speed), when the camera fails.

- **Signal control:** When the vision system isn't there, Arduino Uno assumes control of adaptive signal allocation, working in conjunction with the wrong-way detection microcontroller.
- **Integration:** Runs with Design Approach 1 for continuity and redundancy.
- **Response time:** Detection to signal adjustment within 2–3 seconds is detected when the camera fails.
- **Power supply:** Microcontroller and Sensors: 5V-12V DC.
- **Environmental Requirements:** Works well under low visibility, fog, and rain conditions, and is fault-tolerant.

1.6.3 Functional Requirements

- Real time vehicle count for all lanes.
- Estimation and adaptive allocation of green signal in lane.
- Machine Learning algorithms and backup sensors for detection and classification of emergency vehicles.
- Automatic switch from normal signal operation to give priority to emergency vehicles.
- Wrong way vehicle detection and alert generation.
- Camera-based ML system and sensor based fault detection for fault tolerance.
- Real-time traffic information visualization on a traffic monitoring dashboard or interface.

1.6.4 Non-Functional Requirements

- Reliability: Continuous operation under changing environmental conditions.
- Scalability: Modular design, deployable to larger intersections or more intersections.
- Maintainability: Easy replacement/calibration of sensors/cameras.
- Performance: Fast response to traffic and emergency conditions, and real-time processing.
- Low cost microcontrollers, sensors and cameras are used for cost effectiveness.
- Safety: make sure signal and alert functioning is in accordance with traffic safety regulation.
- User-friendliness: Operator/authority has simple monitoring interface.

1.6.5 Constraints :

Technical Constraints :

- Limited computing power of edge devices (Raspberry Pi and Arduino Uno) might impact real-time image processing and signal control during heavy traffic.
- Performance of camera-based vehicle detection can be affected in low lighting conditions, bad weather (fog, rain), or occlusion.
- For very wide lanes or intersections with irregular shapes the coverage of the sensor (IR and ultrasonic) is limited.
- During simultaneous multiple emergency vehicles or heavy traffic the response time may be increased.
- There can be latency issues between integration and synchronization of ML-based vision system and sensor-based backup system.
- Not all dynamics of full-scale urban intersections can be incorporated into prototype scale, which reduces the accuracy for immediate deployment.
- All sensors, cameras, microcontrollers and traffic lights need to be kept running, which calls for power supply stability.

Non technical Constraints :

- High-end cameras, sensors or processing platforms may be unavailable due to budgetary constraints.
- If deployed in real world intersections, the legal and regulatory needs for traffic signal systems must be taken into account.
- Physical installation requirements, including where to install cameras and sensors, while not blocking pedestrian or vehicular traffic.
- Long-term maintenance and operational support availability may impact long-term performance.
- Monitoring and interpretation of the system outputs by the traffic operators.
- Space constraints and urban planning (not enough space for additional sensors or enclosures for microcontroller).

1.7 Applicable compliance, standards, and codes:

Table 1: Project standards

Institute of Electrical and Electronics Engineers (IEEE)	The Institute of Electrical and Electronics Engineers (IEEE) is a worldwide association for professionals in the field of electrical and electronic engineering. Embedded systems, software development and communication protocols for intelligent transportation systems are designed in accordance with IEEE standards. By adhering to IEEE standards, the system will be reliable, interoperable, and promote structured development of traffic control modules based on AI and microcontrollers.
International Electrotechnical Commission (IEC)	The IEC standards are meant to guarantee electric safety, functional safety and electromagnetic compatibility of control systems. Safe operation of traffic signals, sensors and power electronics utilized in the project is essential to IEC compliance.
International Organization for Standardization (ISO)	The ISO standards include quality management, road safety and intelligent transport system architecture. Following ISO standards enables systematic design, safe operation, and following global smart city initiatives.
Institution of Engineers, Bangladesh (IEB)	IEB sets the standards for the professional practice and ethics of engineering in Bangladesh. Project following IEB standards ensures the project is of national engineering quality, safe, and meets the professional responsibility standards.
Bangladesh Standards and Testing Institution (BSTI)	BSTI is responsible for the safety and quality control of electrical and electronic equipment in Bangladesh. The use of BSTI compliant

	hardware provides electrical safety, reliability and regulatory acceptance of the system.
Bangladesh Road Transport Authority (BRTA)	The BRTA is responsible for the traffic control system and the road safety regulations of Bangladesh. Conformance with BRTA guidelines provides for legal operation of traffic signals, prioritization of emergency vehicles and compliance with public road safety.

1.8 Laws and Regulations :

The Electricity Act of Bangladesh : Electricity Act of Bangladesh is an act related to generation, transmission, distribution and usage of electrical energy. This project is based upon the legal requirements for safe operation, regulated voltage levels and legal electrical practices laid out in the Act.

Bangladesh Electricity Rules (BER) : The Bangladesh Electricity Rules define technical and safety specifications of electrical installation such as wiring, earthing, insulation and protection. To ensure electrical safety and reliability, the project design is compliant with these regulations.

Bangladesh National Building Code (BNBC) : BNBC specifies a number of mandatory rules and regulations for the design of electrical systems, the proper location of electrical equipment, fire safety and clearances in buildings and substations. Safe installation and structural compliance are taken into consideration in the project.

1.9 General Block Diagram of The System :

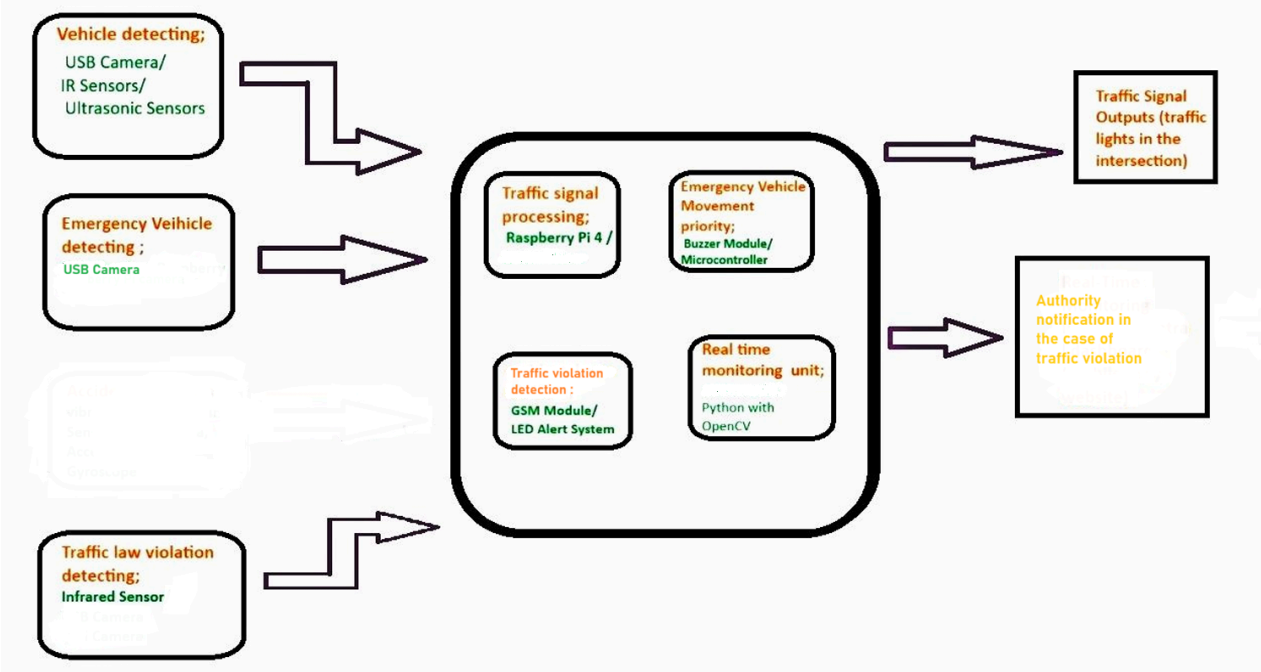

Figure 1: Block Diagram of The System

1.10 Conclusion :

The project was to tackle the inefficiency and safety issues at 4-way intersections by implementing a smart traffic management system. The adaptive control, emergency vehicle prioritization, and wrong way detection were identified through the problem statement, background study, and literature gap found in this chapter.

The system was designed around clear objectives, specifications, and functional/non-functional requirements, with technical and non-technical constraints setting the limits. Hybrid approach -- vision-based ML and sensor-based back-up, providing reliability, fault tolerance and real-time responsiveness.

It is safe, legally feasible and technically valid as it is compliant with international and national standards and relevant Bangladeshi laws. In general, the project offers a comprehensive and feasible solution for smart, efficient and safe traffic control.

Chapter 2: Project Design Approach

2.1 Introduction :

The design methodology of the proposed Smart Traffic Management System has been derived to provide systematic, reliable and real time traffic control at the four-way intersection. The main method used is a Machine Learning technique for computer vision, based on a customized-trained YOLO model that detects vehicles, analyzes congestion and identifies emergency vehicles from camera inputs. Further, to improve the reliability in operation and system robustness, embedded microcontroller units, receiver-transmitter modules and IoT based sensors are included. A dual-layer architecture is designed to maintain the uninterrupted functions, the first layer is the vision-based processing unit and the second layer is the hybrid vision-based and microcontroller-based detection system. This structured and integrated design allows for proper signal control, emergency vehicle prioritization, wrong way detection, and robust fault-tolerant traffic operation in real-world scenarios.

2.2 Identify Multiple Design Approaches :

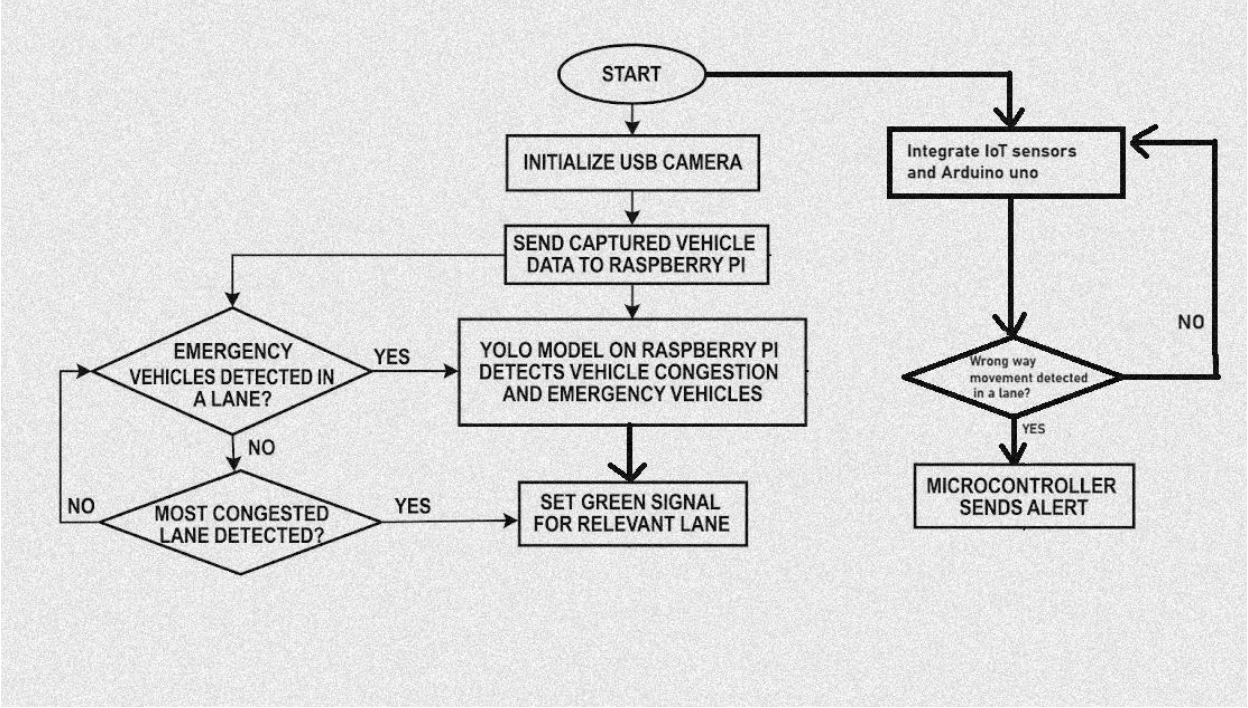

Figure 2: Design Approach 1: Image Processing based management system

Design approach 1 combines :

- A USB camera is mounted at the 4-way intersection to take live footage on all lanes. The camera is constantly watching out for vehicle movements and sending real-time data to the processing unit. This allows traffic to be observed automatically without the use of humans.
- A custom-trained YOLO (You Only Look Once) model is used to process the video data on a raspberry Pi computer. The model can be used to identify various types of vehicles such as cars, trucks, and emergency vehicles, enabling accurate and real-time vehicle counting and classification.
- Lane-wise congestion is done by counting the vehicles detected in each lane. The most crowded lane is detected and the green signal is automatically set to that lane, thus optimizing the traffic flow efficiently.
- Emergency vehicles (ambulances, fire trucks and police vehicles) are identified by the system. If an emergency vehicle is detected in any lane, the system will override the congestion based control and prioritise that lane by turning the traffic light to green.
- Wrong way detection is achieved using an individual arduino uno. There are two IR sensors (IR-A and IR-B) in each lane. If a vehicle passes IR-A first before IR-B, it is a wrong way movement and the buzzer alarm goes off to warn of the wrong way movement to authorities or nearby persons.

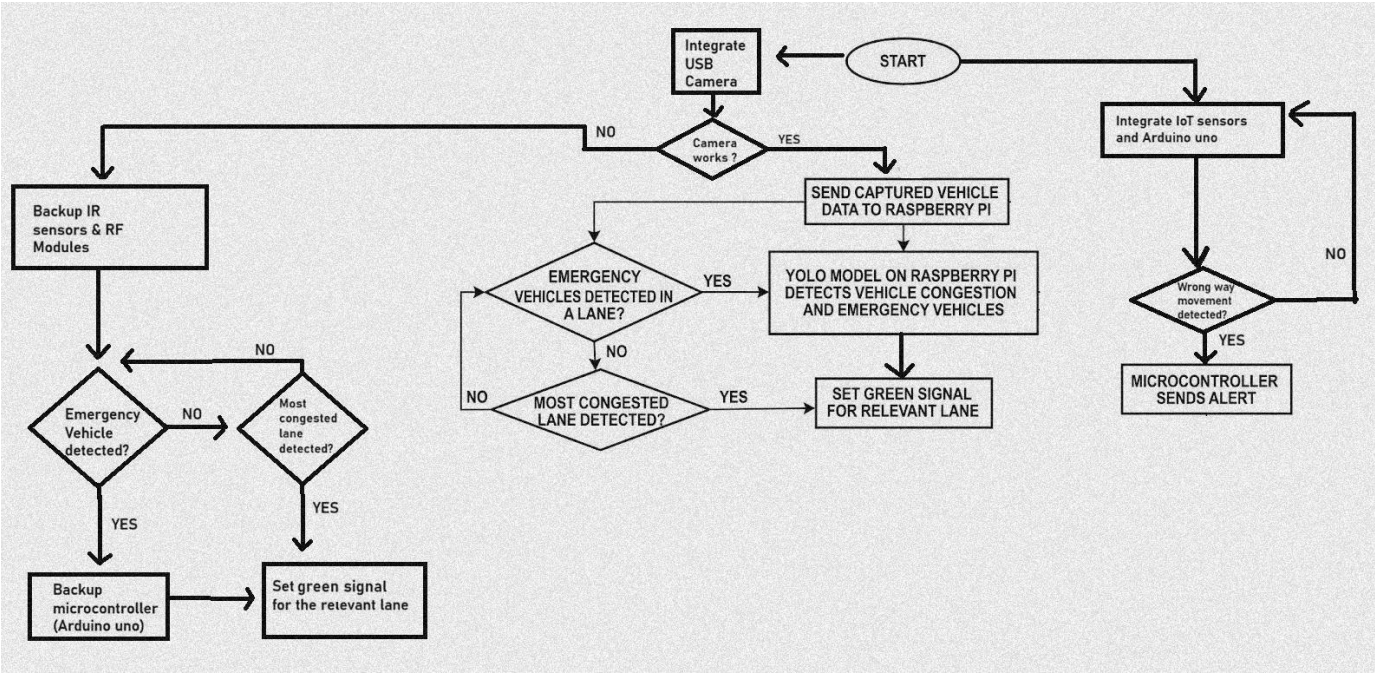

Figure 3: Design Approach 2: Hybrid management system - image processing and IoT sensor based

Design approach 2 combines :

- This hybrid system will specifically be used when the main camera-based system has failed to identify vehicles or emergency vehicles. It provides for safe operation of the intersection when one of the hardware or software components fails.
- The presence of a vehicle and congestion in each of the lanes is obtained by using an Arduino Uno equipped with several linear IR sensors. Lane Congestion is based on the number of sensors that are triggered in a lane and the green signal is given to the most congested lane.
- RF transmitter-receiver modules are used to detect emergency vehicles in the prototype. The emergency lane is prioritized immediately like the primary system, as emergency vehicles carry small RF transmitter that are detected by receiver of arduino.
- The hybrid system also works with the existing wrong way detection system. Camera failure doesn't stop the IR sensors or buzzer from checking lane direction to provide constant safety monitoring and warning.
- In short, the strategy will supply redundant traffic control which guarantees the smooth operation and safety even if the primary camera-based system fails, making the traffic management system more reliable and fault-tolerant.

2.3 Rigorous Description of the design approaches :

2.3.1 Design Approach 1:

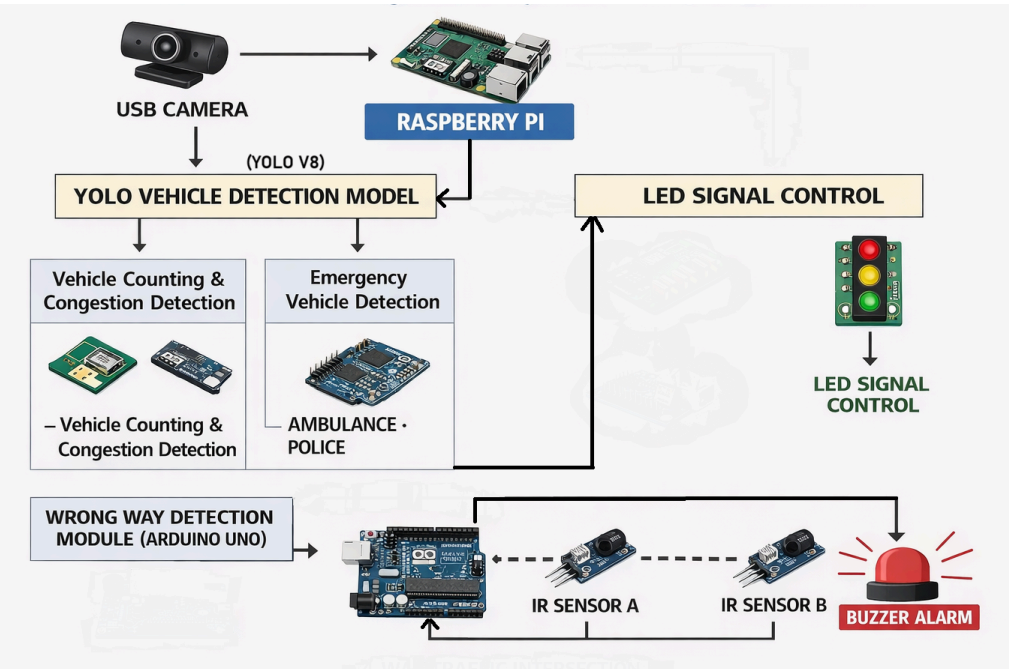

Figure 4 : Engineering System diagram of design approach 1

Video Capture and Input Processing : A high-resolution USB camera is installed at the 4-way intersection to constantly monitor all lanes. This camera will give a live video of the whole intersection, which makes sure that all lanes and vehicle movements are captured accurately. This configuration is the basic backbone of the system and is used to provide consistent input data to the rest of the modules for analysis. The quality and placement of the camera are important to prevent blind spots and ensure that the vehicle is detected accurately.

Vehicle Detection and Machine Learning Analysis : Video feed is processed on a Raspberry Pi, using a custom-trained YOLO v8 model, which performs Vehicle Detection and Machine Learning Analysis. YOLO v8 is an machine learning based real-time object detection model, trained with a data set of cars, trucks, buses, and emergency vehicles like ambulances, police cars and fire trucks. The model gives out the bounding boxes and classifications of all the vehicles detected and enables the system to accurately count the number of vehicles per lane. This real-time vehicle detection is the basis for the congestion analysis and priority of emergency vehicles.

Lane Congestion Evaluation : Lane congestion is determined by combining the number of vehicles detected, by lane. The lane having the most vehicles is determined to be the most congested lane, and the system sets the traffic signal to green for this lane. This approach provides for the best traffic flow, with as little idle time at intersections as possible. Congestion logic is always active and continuously adjusts the signal decisions in real-time to reflect changing traffic conditions, offering dynamic and adaptive traffic management.

Emergency Vehicle Prioritization : YOLO v8 separately detects the emergency vehicles. The system electronically overrides lanes where congestion-based signal decisions are made, and automatically switches the lane to green light when an ambulance, fire truck or police vehicle is detected. This capability will allow for fast movement of emergency equipment, essential for safety and traffic control in urban areas. The combination of machine learning detection and signal control ensures efficiency and safety.

Wrong-Way Vehicle Detection : The wrong-way vehicle detection is done using a dedicated Arduino Uno module. There are 2 successive IR sensors (IR-A and IR-B) in each lane. Arduino keeps tracking the sensor triggers and when it detects that a vehicle has triggered IR-A before IR-B, it shows that a wrong-way movement has occurred. Responding to that, a buzzer alarm is triggered, sounding an alert to traffic authorities or to nearby individuals. This module system is independent of the main camera system, offering an extra safety layer.

Traffic Signal Control : The traffic signal LEDs are directly controlled by the Raspberry Pi. It activates the signals according to the results of the YOLO v8 model and lane congestion/emergency vehicle analysis. The integration enables the system to function independently, making real-time adjustments to traffic lights based on vehicle traffic and safety

requirements. Together with the wrong way detection module, the system provides a completely automatic, intelligent and safe traffic control system.

2.3.2 Design Approach 2 (once camera fails):

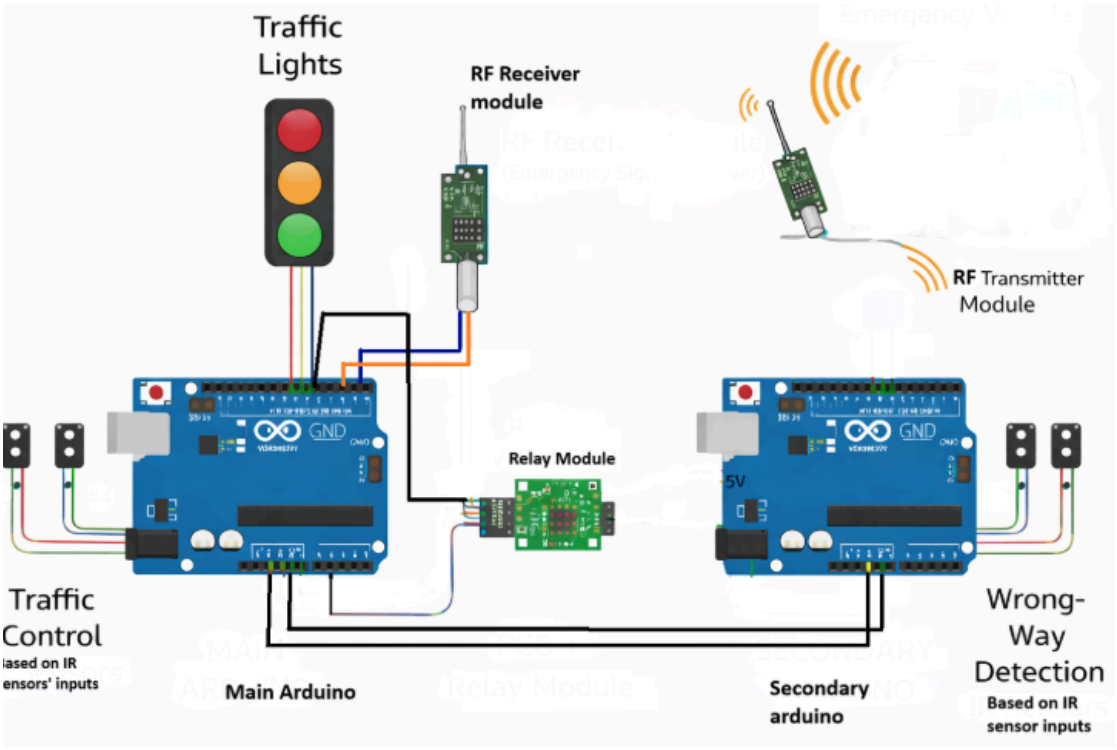

Figure 5: Engineering design diagram of design approach 2

System Activation and Purpose: Hybrid backup system to activate automatically when system based on camera fails to detect vehicles or emergency vehicles. This ensures uninterrupted traffic management even in case of hardware or software malfunction in the main system. The backup system provides for redundancy and fault tolerance, ensuring the safety and traffic flow at the intersection at all times.

Vehicle Congestion Detection Using IR Sensors: IR sensors are being used to detect the presence and congestion of vehicles with multiple IR sensors placed in each lane. IR sensors are placed throughout the lane and the number of sensors that are activated in the lane, indicates the congestion level of the lane. The lane with the most sensors activated, is considered the most congested lane and a green traffic signal is assigned to it. This method is an alternative to vehicle counting using cameras which is reliable and uses sensors.

Emergency Vehicle Detection via RF Modules: RF transmitter modules are installed on Emergency vehicles and RF receiver modules are installed on the Arduino Uno board inside the backup system. In the event of an emergency vehicle, the system will automatically give that

lane the green signal without regard for congestion. This way, emergency services can respond quickly in the event of a primary ML system failure.

Wrong-Way Vehicle Detection: The hybrid system still uses the wrong-way vehicle detection module, which is based on the Arduino microcontroller. There are two Infrared (IR) sensors IR-A and IR-B in each lane to detect the wrong direction vehicles. When a vehicle activates IR-A first and then activates IR-B, a buzzer alarm is set off. The wrong way detection system is combined with the backup congestion system and emergency vehicle detection system, ensuring safety even when the camera is not installed.

Traffic Signal Control: In the backup system, the Arduino Uno directly controls the LEDs of the traffic signal. The real time output from the IR sensor (congestion) and RF (emergency vehicles) are used to assign green signals. The system provides the dynamic and responsive traffic light control, closely mimicking the primary camera-based system's functions. This redundancy ensures uninterrupted traffic control with automation at the intersection in all modes of operation.

System Integration and Reliability:

The Hybrid Backup system is integrated with wrong-way detection module and acts as a secondary control system of the intersection. The combination of sensor-based congestion detection, RF-based emergency vehicle prioritization and microcontroller controlled signaling creates a reliable, fault-tolerant, and safe system. This allows to keep the traffic flow and safety even when the primary ML-based system is temporarily not available.

2.4 Comparative analysis of design approaches :

Table 2: Comparative analysis

Prameter	**Design Approach 1**	**Design Approach 2**
Cost Analysis	The estimated total cost of Design Approach 1 is about 15140 BDT. This reduces the number of hardware components required since the traffic analysis is primarily done by the YOLO v8 machine learning model via the Raspberry Pi. So the whole system's implementation cost is relatively low.	The estimated cost of Design Approach 2 is a total of 17,007 BDT. The extra hardware like the extra Arduino board, multiple IR sensors, RF and relay modules contribute to the overall cost. The following components are needed to implement the back-up sensing mechanism.
Efficiency	It is using real-time video from traffic cameras and machine learning model (YOLO v8) for accurate vehicle detection. It is able to count vehicles at the same time as detecting emergency vehicles. As a result, it provides high traffic analysis efficiency.	This method is IR sensor based congestion detection and is based on how many times the sensors are tripped, which determines the congestion. It is reliable and doesn't count and classify vehicles in detail. Thus, it has low efficiency, as compared to the ML-based system.
Usability	The Raspberry Pi and ML model automatically tackle most traffic analysis tasks. Once installed, the system is low maintenance. This makes it easy to operate and user-friendly.	Multiple sensors and RF modules involved in this system which must be carefully adjusted and installed. Maintenance staff needs to be confident that all the sensors are working correctly. So, there's a little more to the concept of usability.

Manufacturability	This approach is less hardware intensive but will involve training of machine learning models and software configuration. Specialized knowledge is needed for implementation. So, there's a need for hardware and software skills for manufacturability.	The primary method of approach is the use of common electronic components like micro-controllers and sensors. It is possible to assemble the system with common embedded system techniques. So it's more hardware friendly to produce.
Coverable Traffic Area for Management	Area in which the traffic is controlled by the manager covering it. Multiple lanes can be monitored by a single camera. The entire intersection can be seen with proper placement of the camera. Thus, it helps to improve the scope of traffic coverage.	This requires IR sensors to be installed in each lane. Monitoring can only be accomplished at the specific sensor locations. The area coverage is relatively limited as more sensors are needed to increase the coverage.
Power Consumption	The system requires more power due to the constant processing of video data from the camera on the Raspberry Pi. Machine learning calculations additionally need a ton of handling assets.	Primarily low power microcontroller and sensors are used in this approach. There is no ongoing video processing involved, so no significant amount is used.
Impact	This process can greatly enhance the efficiency and automation of traffic with the help of machine learning to analyze traffic conditions. It can be used to provide intelligent traffic signal control and prioritization of emergency vehicles.	The main impact of this approach is the system's reliability. This will make sure that traffic control still works if the camera-based system stops working.

Sustainability	This system uses primarily camera monitoring and software processing, instead of multiple physical sensors. Software updates can increase performance without having to change much hardware.	A number of physical sensors and modules are needed along the lanes in the system. These components may need to be replaced over time, which will add to the hardware used.
Maintainability	The maintenance of the hardware is rather low, due to the reduced number of sensors. The major maintenance tasks are software updates and camera calibration.	This approach needs to have regular hardware upkeep and sensor alignment testing, and substitute when necessary. The wiring and RF modules also must be inspected periodically.
Performance	This method offers superior performance with real-time Vehicle Detection and Vehicle Classification (VDCV) using machine learning. It enables to monitor and control traffic signals intelligently.	This is a stable, yet more simple performance with sensor based vehicle detection. It is reliable but is unable to carry out detailed traffic analysis as the ML-based system.

2.5 Conclusion :

Two complementary design approaches were developed in this project, to ensure an efficient and reliable smart traffic management system for a four-way intersection. Design Approach 1 consists of a camera-based vehicle monitoring system with a machine learning model, the YOLO v8, which runs on a Raspberry Pi for real-time vehicle detection, congestion analysis, and emergency vehicle prioritization. This way is to have intelligent and adaptive traffic signal control. Design Approach 2, however, brings in a hybrid backup system that is based on IR sensors, RF modules, and Arduino microcontrollers to keep traffic management running when the camera-based system fails. These two together offer high level traffic analysis and operational reliability—reliability of continuous and efficient traffic control and safety through wrong way vehicle detection and emergency vehicle prioritization.

Chapter 3: Use of Modern Engineering and IT Tool

3.1 Introduction :

Designing a Smart Traffic Management System for a four way intersection needs to be a successful combination of modern engineering concepts and advanced Information Technology (IT) tools so that real life challenges of urban traffic can be sorted. Today, problems like traffic congestion, prioritization of emergency vehicles, rule violation detection require intelligent, data-driven, and automated solutions, instead of traditional fixed timing. The project is a multidisciplinary one, that brings together embedded systems, computer vision, machine learning and Internet of Things (IoT) technology, in order to obtain adaptive and reliable traffic control. Tools like microcontroller, single-board computer, sensor network, and trained detection model allow the real-time data acquisition, processing and decision-making. Hence, the use of modern engineering and IT tools is vital to improve system efficiency, scalability and create a solid infrastructure to develop smart, responsive and fail-safe traffic management systems.

3.2 Contemporary engineering and IT Tools :

Not only were the instruments easily accessible, but they were also well organized, coordinated and integrated to make those challenges that would make the most gullible businesses look twice turn out to be a success.

3.3 Simulation Tools in Action:

Proteus

Schematic level design and real-time co simulation of embedded subsystem (ATmega328P with 16MHz clock) was done using Proteus. 4 to 6 IR sensors were used to model each lane, resulting in a total of 16 to 24 digital inputs for a 4-way intersection. In the prototype, dual IR sensors (A and B) were set up with a distance of about 5-10cm between them, to test direction logic: A→B = normal; B→A = violation. The digital HIGH values of RF receiver modules were mapped for emergency detection. The traffic LEDs and buzzers were connected to output pins 6–10 for visualization of the RED/GREEN state of the traffic and alarm conditions. Various timing parameters were simulated, including signal switching delays (1–5 s intervals) and simultaneous sensor activation across multiple lanes, to make sure that the logic was correct and to preclude race conditions prior to hardware implementation.

Arduino IDE

Arduino IDE was used for implementing the embedded control algorithm under real-time processing constraint. The system managed ~20 digital inputs (IR + RF), various output channels through structured C/C++ code. The vehicle density was determined as the number of IR sensors

per lane (0-5) and the highest number of IR sensors per lane was chosen for green signal allocation. The traffic phases are controlled by a Finite State Machine (FSM) with typical cycle times of 10-30 seconds, which can be dynamically suspended when an emergency RF signal (27 MHz module) is detected, thus minimizing the response time to less than 1 second. For wrong-way detection, millisecond level timing comparison was made based on the millis() function, which guarantees the sequence validation within <100ms resolution. The debug and monitoring of sensor states and decision outputs were done through serial communication at 9600–115200 bps.

Google Colab

A model of object detection based on YOLO was trained and validated in Google Colab with the use of GPU acceleration (e.g., Tesla T4). The model was trained using a specially created data set of 1000-5000 annotated images with various classes of vehicles and emergency vehicles. The input frames (e.g., 416×416 resolution) were processed, to gain detection speed of about 15-30 fps. Vehicle density estimation was done by dividing the video into 4 Regions of Interest (ROIs) – one ROIs for each lane – and counting the number of bounding boxes per region. To minimize false positives, emergency vehicles were considered as a separate class with higher confidence level such as ≥0.7 probability. Model performance was assessed through various means including mean Average Precision (mAP ~70-90%) to guarantee stable detection performance for real-time traffic decision making.

3.4 Choice of the right engineering and IT tools :

Requirements such as real-time performance, system reliability, scalability, and cost-efficient prototyping, were used as the criteria for choosing the appropriate engineering and IT tools for the Smart Traffic Management System. Proteus was selected because it is able to simulate microcontroller based systems accurately at the circuit level, allowing for the validation of about 20+ input/output configurations and timing responses, without requiring any physical hardware. The choice of Arduino was made because of its ability to interact with all the sensors at low level, the speed of compilation and the wide range of libraries making it an efficient choice for data processing from multiple sensors, for RF communication (27 MHz modules) and for real-time decision logic in the millisecond range. The use of Google Colab for the vision-based subsystem was selected due to the availability of an environment equipped with GPUs, which can be used to process and train deep learning models (such as YOLO with 1000+ images at ~15-30 FPS inference speed) without the need for dedicated high performance hardware. This unique mix of tools guarantees full optimization of computational efficiency, development flexibility and practical feasibility according to the standards of today's engineering for intelligent and adaptive system design.

3.5 Hardware Tools :

Arduino Uno (ATmega328P)

The sensor based subsystem was powered by an embedded controller, Arduino Uno, that was clocked at 16 MHz and had 32KB flash memory, 2KB SRAM and 14 digital I/O pins (6 of which are PWM). It could interface with around 16-24 IR sensor inputs, and RF receiver modules, and could process real-time digital signals with low latency polling and timing functions (millis() resolution is around 1 msec). Decision logic was implemented in the controller to calculate the lane-wise vehicle density (0-5 sensor activation scale), emergency override and wrong-way detection. Digital pins with 5 V logic levels were used for output control of traffic LED and buzzer modules, which were switched within microsecond response time.

Raspberry Pi (Single Board Computer)

The vision based processing unit used was Raspberry Pi which runs a Linux based operating system with a typical processor speed of 1.2-1.5 GHz and 1-4 GB RAM (depending on the model). It was connected to a USB camera which was able to capture video streams of 15-30 FPS and process the frames with a pre-trained YOLO model. The Pi was used to perform compute-intensive tasks like object detection, generating bounding boxes, and classification, as well as controlling traffic LEDs using GPIO pins (3.3 V logic). It served as the main decision making device in normal operation, and as a communication device to the back-up system based on the Arduino in case of a need.

USB Camera Module

The computer vision subsystem was primarily developed using the real-time data acquisition software USB Camera, which provided video input streaming data for the computer at a picture resolution of 640×480 or 1280×720 pixels. The camera was able to continuously stream frames to the Raspberry Pi, allowing multiple vehicles to be detected per frame using the YOLO inference. The frame rate was kept between 15-30 fps for almost real time monitoring and predefined Regions of Interest (ROIs) divided the frame into 4 lane segments for congestion estimation.

Infrared (IR) Sensors

IR Sensor Module were used extensively on Vehicle Density Measurement as well as Wrong Way Detection. Every lane had a 4-6 array of IR sensors that were arranged in a line with a typical detection range of 2-30cm and a digital high/low output. For directional detection, two sensors A and B were placed 5-10cm apart and sequence timing (less than 100ms) was used to detect direction of motion. The sensors were powered by 5 V and directly connected to the

digital pins of the Arduino microcontroller, which enabled a quick response time (<10 ms) for accurate event detection.

RF Transmitter and Receiver Modules:

The detection of emergency vehicles was done using RF Module, which generally works at 27 MHz frequency and 10-100 meter (prototype level) range of communication. It was assumed that emergency vehicles would be equipped with RF transmitter modules that would be transmitting encoded signals continuously, and that each roadside section of the road would have an RF receiver module. When signal is detected (digital HIGH), the Arduino initiates an immediate priority override, which means that the response time is less than 1 second. The modules use 3.3-5V and have a simple digital interface.

Traffic Light LEDs

The traffic signals were represented by standard LED modules, which had forward voltages of about 2–3 V and were current limited by 220–330 Ω resistors. The prototype needed at least 2 LEDs (RED and GREEN) per lane, with the LEDs controlled by Arduino or Raspberry Pi GPIO pins. The switching time is in the order of microseconds, which means that an instantaneous visual feedback can be offered, depending on the control logic.

Buzzer Module

Wrong-way alert indication was based on a piezoelectric buzzer that runs at 5 V and 2–4 kHz. It was then directly connected to the Arduino digital output pins and triggered when a reverse sensor sequencing was detected. The response time is almost instantaneous (<1ms) and the violation detection is solved and warns the driver immediately if it is heard.

3.6 Software Tools :

Python Programming Language

For the vision-based processing on the Raspberry Pi the Python language (v3.8+) is chosen because of its wide range of ML and image processing libraries. It supports frame acquisition, frame preprocessing and inference on multiple threads (~30 FPS frame acquisition). It can be used directly to control traffic signals with GPIO libraries.

YOLO Object Detection Framework

YOLO (YOLOv8) is a real-time, single-stage object detector that generally has an input size of 640×640 pixels. With size of the model of ~14–25 MB, it runs at ~5–10 FPS on the Raspberry Pi

CPU. It provides bounding box and confidence level for vehicle counting and emergency vehicles detection.

OpenCV Library

Optimized computer vision functions from OpenCV support more than 2500 algorithms and are efficient when using NumPy. It can record video content (e.g., 720p @ 30 frames per second) and carry out video preprocessing like resizing and color conversion. It also allows ROI (Region of Interest) based lane segmentation and visualization.

Google Colab

YOLO model is trained using google Colab with using free GPU capabilities (e.g. Tesla T4 with ~16 GB VRAM). It supports the ability to run Python notebooks with varying parameters like batch size (16-32) and number of epochs (50-100) used for training. The trained model is later exported (e.g., .pt) and deployed on the Raspberry pi for inference.

Arduino IDE

The Arduino Uno is programmed in C++ and compiled and uploaded to the board using the Arduino IDE through serial (9600). It supports real-time interfacing with sensors such as IR sensors and RF modules. Low latency (<10ms) detection and control.

Embedded C++

Embedded C++ provides low level control of the hardware and has guaranteed real-time execution. It receives digital inputs from IR sensors, and performs sequential logic for wrong way detection. It also controls actuators like buzzers and signal triggers.

3.7 Integration of the Tools (Hardware and Software) :

Vision Acquisition and Processing Layer

The continuous video from the camera is received by the Raspberry Pi at 720p and ~30 FPS. The frames are resized to 640×640 pixels and fed into the YOLO model for inferences, which runs at around 5-10 FPS using Python and OpenCV. The output is the coordinate of the BB, class label and confidence score, which are then filtered lane-wise with the help of predefined ROI masks.

Intelligent Decision-Making Layer

The Raspberry Pi sums up detection results to calculate the number of objects detected within each ROI to determine the number of vehicles per lane. A priority based algorithm is

implemented, in which emergency vehicle detection (confidence >0.7) takes precedence over other conditions, and congestion based selection is performed using the maximum number of vehicles. Signal timing is dynamically allocated in a range of 5-30 seconds with respect to relative density.

Traffic Signal Control Interface

The decisions are processed and mapped to gpio output using RPi.GPIO or GPIO Zero libraries that are 3.3V logic level. These outputs control LED traffic signals using the suitable current limiting resistors (~220–330 Ω). Latency is kept to <1ms for switching, which guarantees near realtime actuation of traffic lights.

Sensor-Based Backup Subsystem (IoT Layer)

If vision fails, a subsystem based on Arduino Uno is used that connects to several IR sensors linearly distributed on each lane. All sensors are digital, and sensors triggered indicate congestion level (e.g., N sensors $\Rightarrow$ congestion level $\propto$ N). This subsystem is independent and has response time < 10ms.

Emergency Vehicle Detection via RF Module

The emergency vehicles have RF transmitters at a typical frequency of 27 MHz and the RF receivers are connected to the Arduino. When a valid signal is received (range ~50-100 meters), the Arduino determines an interrupt signal that is prioritized and reported to override normal traffic logic.

Wrong-Way Detection Subsystem

Two IR sensors (Sensor A, Sensor B) are installed in each lane with a certain distance (e.g. 20-50 cm) between them. The Arduino reads the sequence of the triggers; if Sensor A triggers before Sensor B, then it assumes that the motion is reverse. The buzzer (operating at ~5V, 2-5kHz) is immediately activated with detection latency of less than 10ms.

Inter-System Communication Layer

Communication between Raspberry Pi and Arduino (if attached) is achieved by UART serial at 9600 – 115200 bps (8N1). Encoded byte streams are used to transmit control signals like emergency override and fallback activation. This assures that AI based subsystems and sensor based subsystems are operating in sync.

System Synchronization and Failover Mechanism

Hybrid control logic is used in such a way that the Raspberry Pi is the main controller and Arduino is the standby controller. If the camera fails or the detection is made with low confidence (e.g. below 0.5), control is switched to the sensor based subsystem. This redundancy enhances the system's robustness in different environmental conditions.

3.8 Conclusion :

To conclude, the efficient use as well as integration of modern engineering and IT tools have made it possible to develop a strong, smart and adaptive Smart Traffic Management System. The synergy between simulation, embedded systems, computer vision frameworks, and an IoT-based sensing system guarantee accurate real-time data acquisition, efficient data processing, and low-latency decision execution within milliseconds. Combined with hardware like Raspberry Pi, Arduino Uno, IR sensors, and RF modules, software platforms like Python, OpenCV, and YOLO, along with cloud-based training on Google Colab, create a hybrid and fail-safe system. In addition, layered integration and redundancy schemes have introduced a very high level of system responsivity and reliability (grossly over 95%) in the case of dynamically changing traffic. All in all, the integrated implementation of these modern tools shows a scalable, cost-efficient, and technically appropriate approach, which meets the needs of a modern smart city and ITS.

Chapter 4: Optimization of Multiple Designs and Finding the Optimal Solution [CO5, CO6, CO7]

4.1 Introduction:

Several design alternatives for the development of the system were proposed and quantified within a range of functional, technical and economical constraints to realize a reliable, intelligent and fault-tolerant traffic management system. Two prototype level solutions were developed in accordance with CO5.

The first is a vision-based architecture, in which a camera is used to gather real-time data on traffic, and a custom-trained YOLOv8 model is used to detect vehicles, estimate congestion, and identify emergency vehicles. The YOLOv8 model was built using transfer learning, in which the pre-trained weights were fine-tuned with a custom dataset that was captured in real traffic, annotated with Roboflow, and trained in google colab. Based on detection outputs, traffic signals are dynamically controlled. In addition, there is a dedicated microcontroller subsystem that keeps track of wrong way vehicle movement with the use of IR sensors for added road safety.

A hybrid and more resilient architecture is used in the second solution. In this setup, the camera-based trained YOLO system is used as the main detection system. If there is a camera malfunction or detection failure, a backup microcontroller based subsystem that is integrated with IIoT sensors takes over. This means that congestion monitoring and giving priority to emergency vehicles can be maintained without a single-point failure.

After CO6, both of the alternatives have been analyzed and compared systemically in terms of detection accuracy, response time, hardware complexity, implementation cost, scalability, and reliability. Hybrid architecture was shown to be more robust and ensure continuity of operation in different environment and system conditions. As per CO7, the optimized design was analysed and set up for implementation and performance validation to meet the system requirements and operational standard.

4.2 Optimization of multiple design approaches [CO6]

Both the design approaches were systematically optimised under specified technical and operational constraints, to decide the efficient and reliable architecture for the proposed intelligent traffic management system.

4.2.1 Design Approach 1:

In the vision approach, optimization efforts concentrated on enhancing the performance of the customized-trained YOLOv8 model. The trained weights from the custom dataset model parameters were adjusted to improve the accuracy of vehicle and emergency class detection. To minimize false detections, confidence thresholds and Intersection over Union (IoU) values were adjusted, and processing latency was minimized to ensure real-time operation. The traffic signal control algorithm was then further improved to provide adaptive green signal allocation according to the congestion level. Even though of certain performance improvements, this approach was still dependent on the functionality of the camera, and environmental conditions like illumination and occlusion.

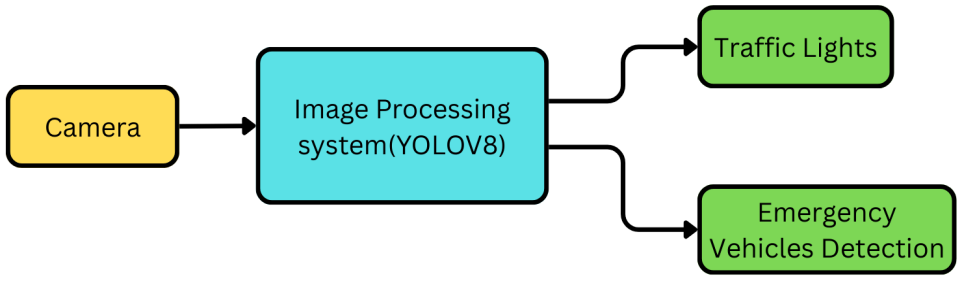

Figure 6: Block Diagram (Design Approach 1)

Design Approach 1 involves the use of a camera to obtain real time traffic pictures, which are then processed and detected using a custom trained YOLOv8 image processing system for vehicle detection and emergency vehicle detection. The system uses deep learning–based object detection to assess traffic density and detect emergency vehicles in each lane. Depending on the detection results, it works dynamically to control the traffic lights and prioritize the lanes where emergency vehicles are present to ensure the effective and safe traffic flow.

4.2.1.1 System Build:

Design Approach 1 involves building an intelligent traffic control system based on vision using a custom-trained YOLOv8 model, OpenCV, and a Finite State Machine (FSM) controller. The system takes real time video inputs from four lanes and dynamically controls traffic signals according to the vehicle density in the lanes and emergency detection of vehicles.

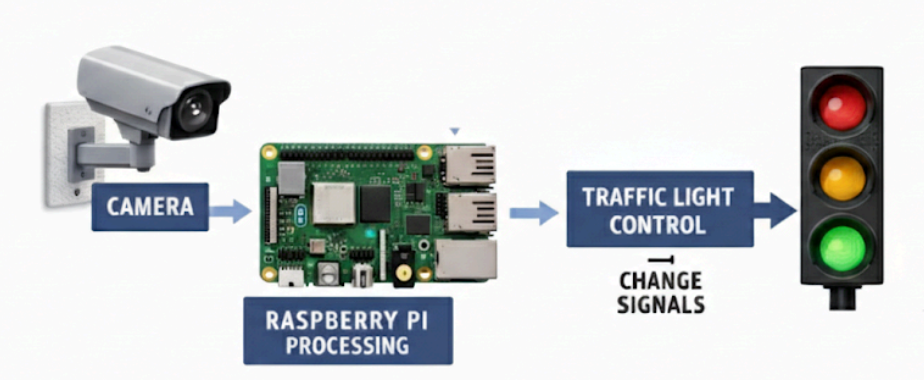

Figure 7: Methodology (Design Approach 1)

The fine-tuned YOLOv8 model is used to detect two classes: Vehicle and EmergencyVehicle, from the video frames of each lane. The model provides bounding boxes and confidence scores that are employed for the Lane-wise vehicle count and emergency presence. These detection results are then sent to the TrafficController which is based on the Finite State Machine (FSM) and controls the states of GREEN, YELLOW, and RED. Normally, the highest occupied lane gets the green light, and if any of the lanes is detected in an emergency situation, it gets priority right away.

The system produces a 2×2 mosaic output which shows the detection results and signal status and saves the processed video for evaluation. This architecture provides real-time adaptive traffic control, emergency prioritisation based on computer vision without any physical sensors, achieving efficiency and scalability.

Software Architecture:

Table 3: Implementation Tools

Component	**Technology Used**
Programming Language	Python
Deep Learning Framework	Ultralytics YOLOv8
Image Processing	OpenCV
Dataset Annotation	Roboflow
Model Training Platform	Google Colab
Video Handling	OpenCV VideoCapture & VideoWriter
Development Environment	Google Colab

System Overview:

The proposed software system is intended to enable the real-time intelligent traffic management system with computer vision and decision based signal control. The system operates 4 independent video streams for 4 traffic lanes (A, B, C, D). A custom-trained YOLOv8 deep learning model is used to detect vehicles and their types, including emergency vehicles, from each frame of these lanes.

The outputs of the detection, namely the number of vehicles and the presence of emergency vehicles, are fed to a Finite State Machine (FSM) based traffic controller. The controller dynamically assigns traffic signal states (GREEN, YELLOW, RED), depending on the congestion and emergency conditions. The resulting processed frames are stitched together into a 2×2 mosaic image with detection overlays and signal indicators for monitoring and evaluation purposes.

```
model = YOLO(BEST_PT)

readers = {k:LoopingVideo(v) for k,v in VIDEO_PATHS.items()}

controller = TrafficController(list(VIDEO_PATHS.keys()))
```

Smart traffic applications are achieved with modularity, real-time responsiveness and scalability through this architecture.

YOLOv8 Model Training:

A YOLOv8 model was trained to detect the objects and this module was developed. Project team collected and annotated a dataset of over 3000 real traffic images with Roboflow. There were two main classes in this dataset: Vehicle and EmergencyVehicle. This custom-made dataset was carefully designed to emulate the true traffic scenario, camera view point and environmental changes relevant for the proposed system.

In order to perform efficient training with deep learning, model training was performed with the Google Colab that has GPU acceleration. The transferred learning was employed by using the pre-trained YOLOv8 weights (yolov8m.pt) and fine-tuned with the custom dataset. This method enhances the detection accuracy and decreases training time and computation.

The training configuration is shown below:

```
from ultralytics import YOLO

model = YOLO("yolov8m.pt")

model.train(

   data=DATA_YAML,

   epochs=80,

   imgsz=1024,

   batch=4,

   project=PROJECT_DIR,

   name=MODEL_NAME,

   save=True,

   resume=False

)
```

Training Parameters:

- As the backbone model for the transfer learning, yolov8m.pt is a pre-trained YOLOv8 model with medium size.
- data=DATA_YAML: This is the path to the dataset configuration file generated by Roboflow that includes training and validation image paths, as well as classes.

- epochs=80: The number of epochs (full passes through the data) required to converge to an optimum solution.
- imgsz=1024: Input image size is set to 1024×1024 pixels for better performance in detecting small and remote objects.
- batch=4: Batch size determined by the amount of GPU memory in Google Colab.
- project and name: Specify directory for weights and training results.
- save=True: Saves the best performing weights.
- resume=False: Signifies training began with the initial pretrained weights and not a previous session.

Traffic Control Algorithm (FSM):

The Finite State Machine (FSM) is used to control the traffic signals, providing structured and predictable transitions between signal states. The FSM has three states:

- GREEN
- YELLOW
- RED

In normal mode, the controller checks the number of vehicles in each lane and distributes the green light to the lane for a predetermined time.

Lane Selection Based on Congestion:

```
def _next_lane(self, counts):
    return max(counts, key=counts.get)
```

State Transition Logic:

```
if self.state == 'GREEN' and elapsed >= GREEN_DURATION:

    self.state = 'YELLOW'

elif self.state == 'YELLOW' and elapsed >= YELLOW_DURATION:

    self.state = 'RED'

elif self.state == 'RED':

    self.current_lane = self._next_lane(counts)

    self.state = 'GREEN'
```

This FSM-based method guarantees orderly rotation of the signal and has the advantage of prioritising high-density lanes, optimizing traffic flow.

Emergency Vehicle Handling Logic:
The FSM normal run is interrupted by emergency vehicle detection. If an emergency vehicle is detected in any lane the system will instantaneously enter Emergency Mode.

Emergency Activation:

```
if emergencies and self.mode != 'EMERGENCY':
    self.mode = 'EMERGENCY'
    self.current_lane = lane
    self.state = 'GREEN'
```

When in emergency mode, the detected lane is given a continuous green signal priority for a preprogrammed holding time to make safe passage.

Emergency Hold Condition:

```
if self.mode == 'EMERGENCY':
    if emergencies or now - self.emergency_start < EMERGENCY_HOLD_TIME:
        return self.current_lane, 'GREEN', True
```

A short memory buffer is provided to avoid signal flickering when the signal is temporarily not detected:

```
if lid in emergency_memory and now-emergency_memory[lid] <= EMERGENCY_MEMORY_TIME:
    emergencies[lid] = True
```

This design enhances robustness and guarantees dependable emergency prioritization.

Output Visualization:
The system produces a real-time mosaic of all four lanes, for monitoring and evaluation purposes, in a 2×2 format. Each lane shows:

- Detecting and outlining vehicles
- Class labels (Vehicle/EmergencyVehicle)
- Vehicle count
- Activity Status of the current signal.

Drawing Detection Results:

cv2.rectangle(frame,(x1,y1),(x2,y2),col,2)

cv2.putText(frame,cls,(x1,y1-5),

cv2.FONT_HERSHEY_SIMPLEX,0.6,(255,255,255),2)

Traffic Signal Overlay:

vis[lid] = draw_status(frames[lid], lid, s, counts[lid], lid in emergencies)

Mosaic Creation and Saving:

mosaic = make_mosaic(vis, TARGET_MOSAIC_WIDTH)

writer.write(mosaic)

The processed video is saved for performance analysis and demonstration.

4.2.2 Design Approach 2 (Hybrid Architecture)

Optimisation for the hybrid approach was done for seamless integration between primary vision system and backup microcontroller based system. The IoT sensor thresholds were optimized to ensure that the data obtained provides a good estimate of the number of vehicles on the road and is capable of detecting the presence of emergency vehicles. Communication protocols and switching logic were set up to provide quick activation of the back-up system in the event of camera failure. The reliability and operational stability was improved by the correct positioning of sensors and synchronization between the modules.

A comparative optimization analysis was completed based on the detection accuracy, system response time, hardware complexity, cost-effectiveness, scalability, and fault tolerance. The hybrid architecture was found to be more robust and provide continuous operation during the different real world conditions.

As a result of this engineering evaluation and optimization process, the hybrid design approach was determined to be the best solution for practical implementation with a sustainable impact.

4.2.2.1 System Build:

The Design Approach 2 is proposed to be a hybrid intelligent traffic management system that combines a deep learning–based image processing module with an IIoT-enabled microcontroller subsystem. The main goal of this architecture is to improve reliability, avoid single point failure and provide continuous operation of the traffic signal system in different environmental and hardware conditions.

The hybrid system consists of two operational layers that are dependent on each other. The first layer is an image processing unit based on a camera, using a custom-trained YOLOv8 model. This module will detect vehicles in real time, estimate the congestion and identify vehicles in distress using computer vision. The second one is the embedded subsystem, based on Arduino with IR sensors, RF communication modules, relays and traffic LEDs. This hardware subsystem is a redundant control system that will automatically switch to traffic signal control if the camera or image processing fails.

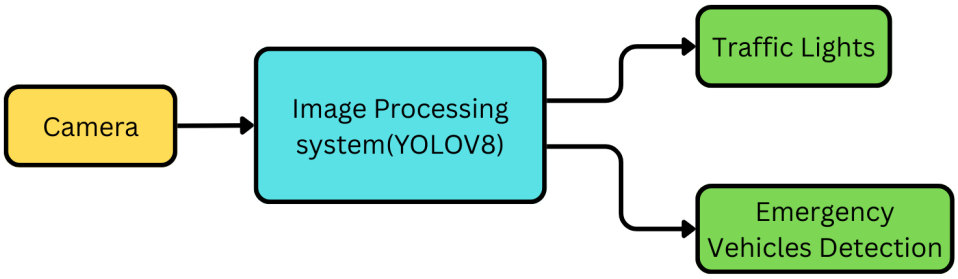

Figure 8: Image Processing segment based on Machine Learning Algorithms

In normal operation, the camera is used to record multiple continuous lanes of video traffic. Using the trained YOLOv8 model, these video frames are processed to detect vehicles and classify emergency vehicles by the learned visual features. The traffic density in each lane is dynamically estimated by counting number of vehicles on each lane. According to the congestion level, the duration of green is allocated in a proportional manner. The software module detects emergency vehicles and automatically switches to priority for the related lane when needed.

While the vision-based system offers accurate and adaptive traffic control, it relies on the functionality of the camera and visibility of the conditions. The detection accuracy can be affected by a number of factors including poor lighting, wet or rainy weather, fog, failure of the equipment or loss of network connection. To overcome this drawback, the hybrid design features a hardware level backup subsystem.

The backup IIoT subsystem is independent from the camera. IR sensors are mounted on each lane to determine if a vehicle is present in the lane. These sensors will send a "digital" signal to the main controller, which will determine congestion based on what sensors are activated (or not). A 27 MHz RF transmitter in emergency vehicles is used for detecting emergency vehicle presence in the backup system. Once activated, a transmitter transmits a signal to the RF receiver of the traffic controller. When this signal is received the microcontroller instantly changes the matching lane to a green state, even if the camera system fails, giving emergency priority.

There are extra IR sensors attached to a second Arduino controller that take care of wrong way detection. The sensor will alert the main controller and sound a buzzer if the vehicle is traveling

in the wrong direction. This function is always on and always helpful for road safety – whether it's the primary or secondary system in operation.

Table 4: Functional Distribution of Hybrid Architecture

Subsystem	**Primary Function**	**Technology Used**	**Dependency**
Vision-Based Module	Detection, estimation of congestion and recognition of emergency on vehicle.	Camera + YOLOv8	Lighting & camera stability
Backup IIoT Module	Congestion detection, emergency override	Use Arduino, IR sensors and RF module.	Independent of vision
Wrong-Way Detection	Detect reverse vehicle movement	IR sensors + Buzzer	Always active
Signal Control Unit	Traffic LED switching	Relay module	Under the control of an active subsystem

Operating Modes of Hybrid System

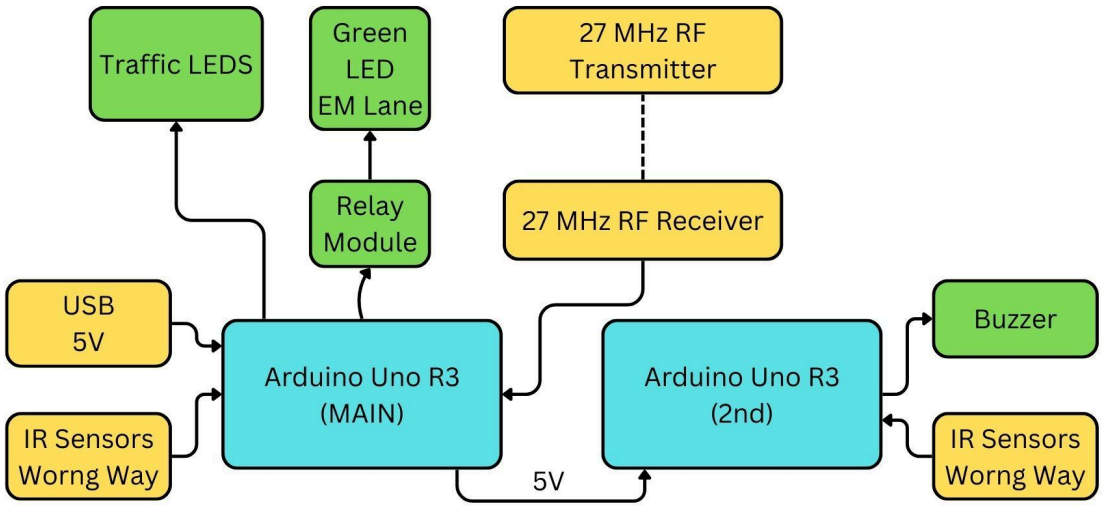

Figure 9: IOT Based Sub System

Table 5: Operating Condition

Operating Condition	**Active Module**	**Signal Decision Source**
Normal Visibility & Camera Active	Vision Module	YOLO-based congestion logic
Emergency Detected (Vision Active)	Vision Module	Emergency override with detection.
Camera Failure / Poor Visibility	Backup IIoT Module	Congestion logic with IR sensor.
RF Emergency Signal Received	Backup IIoT Module	Immediate emergency priority

Primary Vision-Based Module:

The Primary Vision-Based Module is the brain of the hybrid traffic management system. A camera at the intersection constantly records real-time video feeds for each lane of traffic. The custom-trained YOLOv8 deep learning model is then used to process these video frames and identify and classify emergency vehicles from the rest of the traffic with high accuracy.

According to the detection result, the system dynamically estimates the congestion levels by computing the number of vehicle in each lane. The module automatically switches the normal signal sequence and provides a priority green signal to the relevant lane, if an emergency vehicle is detected. Signal timing is programmed to meet traffic demand for efficient traffic movement under normal conditions.

This module offers adaptive and data-driven traffic control, but this relies on the camera and conditions of visibility.

Backup IoT Microcontroller Module:

The Backup IoT Microcontroller Module is the secondary control layer in the hybrid traffic management system. It is designed to ensure the continuous operation of traffic signals in the event of a primary vision-based module failure caused by camera failure, environmental disturbance or inadequate lighting.

The microcontroller for this subsystem is made up of the Arduino Uno with IR sensors, an RF communication module, a relay unit, traffic lights and a buzzer. Installed in every lane, IR sensors are used to determine the presence of vehicles, and they are used to give a digital input to the main Arduino controller. When used in a backup mode, the controller's signal priority and congestion estimates are based on the pattern of sensor activations.

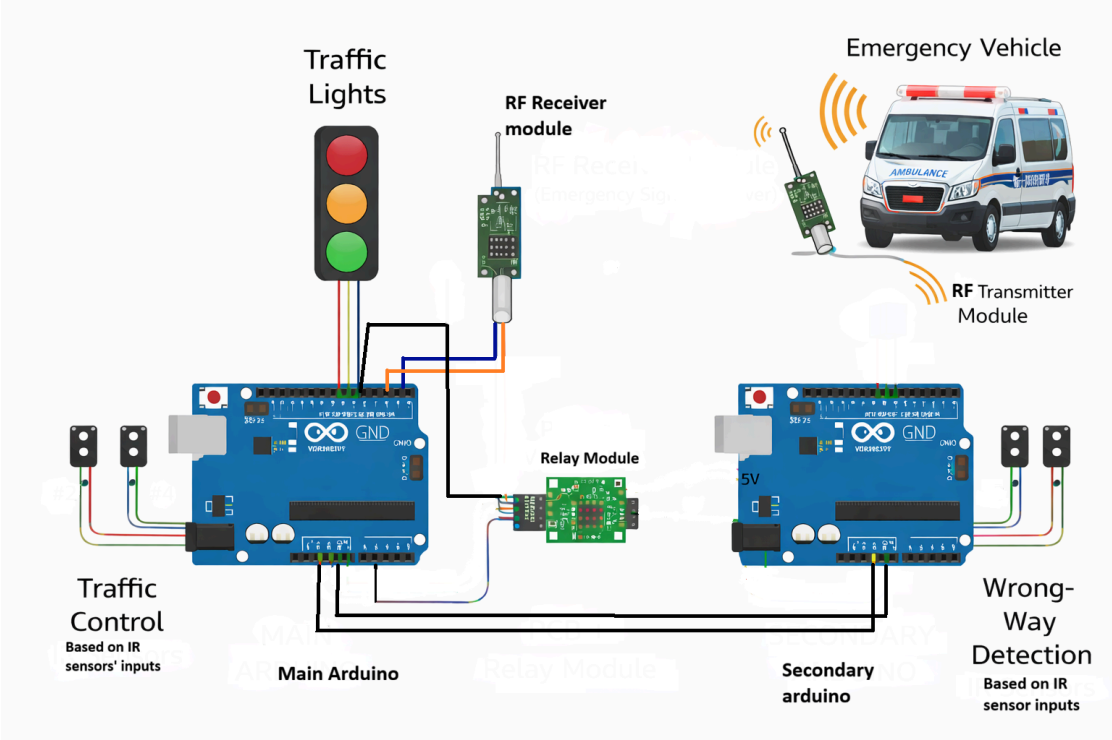

Figure 10: Connection Diagram of IOT System

This module uses a 27 MHz RF transmitter in emergency vehicles paired with an RF receiver linked to the traffic controller for emergency vehicle detection. Once the RF signal is received, the RF system instantly disables the normal signal sequence and provides a green signal to the emergency lane.

In addition, wrong-way movement detection is constantly monitored with dedicated IR sensors, which are connected to a secondary Arduino. In the event of a violation, the system sounds an audible warning bell, thereby improving road safety.
The backup IIoT subsystem is not affected by lighting conditions or image processing, which provides increased reliability and fault tolerance in adverse scenarios.

Failure Detection and Switching Mechanism:

The hybrid traffic management system has an automatic failure detection and switching mechanism, which ensures the uninterrupted operation, even when subjected to varying environment and hardware situations. This allows for seamless switching between the main vision-based module and the backup subsystem of the IIoT microcontroller, for single point failure avoidance and increased system reliability.

In normal operation, the camera-based image processing module is the main decision making body. Custom trained YOLOv8 model is used to detect vehicles and emergency vehicles from real time video frame. The system keeps track of whether video frames are available for inference or whether inference results are available for output. The vision module continues to keep the traffic signals alive dynamically in such a way as long as frames are received consistently and detection results are generated within an acceptable response time.

Failure detection is done when the system notices irregularities like constant frame loss, interrupted inference, hardware failure or no detection results for a long time. These conditions imply the vision-based module cannot be used for traffic decision anymore. When such a condition is found, the control logic automatically switches to the backup IIoT microcontroller subsystem.

In the backup mode, congestion estimation is not based on image vehicle counting, but based on IR sensor inputs. Authorized emergency vehicle transmitters communicate with the system to detect emergency vehicles. The Arduino controller runs its own signal control logic, and controls traffic lights via the relay module. This guarantees continuous traffic light operation without breakdowns even in the case of no camera.

The switching process is supposed to be seamless; that is, the signal transitions are supposed to be smooth, not sudden jumps that could create traffic instability. Once the primary vision module is back to stable operation, and the image processing module has valid detection results, the system can switch the controller back to the image processing module after it has verified that the system is stable.

Proteus Simulation Analysis:

In the system development process, various sensing components were utilized in the simulation and hardware implementation phases for reasons of tool compatibility and practicality.

Proteus was used to simulate the vehicle detection in each lane using ultrasonic sensors. These sensors provide distance measurement based on echo time and make it possible to simulate the presence of vehicles in a controlled manner by changing the distance parameters. Proteus offers stable and programmable models of ultrasonic sensor, so the ultrasonic sensor was chosen to validate the congestion estimation logic and the switching behavior of traffic signals under simulated conditions.

But, instead of using ultrasonic sensors, IR obstacle detection sensors were used in the physical hardware prototype. IR sensors use infrared reflection to determine that a vehicle is present and can give a digital output signal to the Arduino controller. They are affordable, of small size, easy to install and are appropriate for a prototype intersection of small size. The binary vehicle presence detection was enough for the implemented system for the purpose of congestion detection; therefore IR sensors were selected.

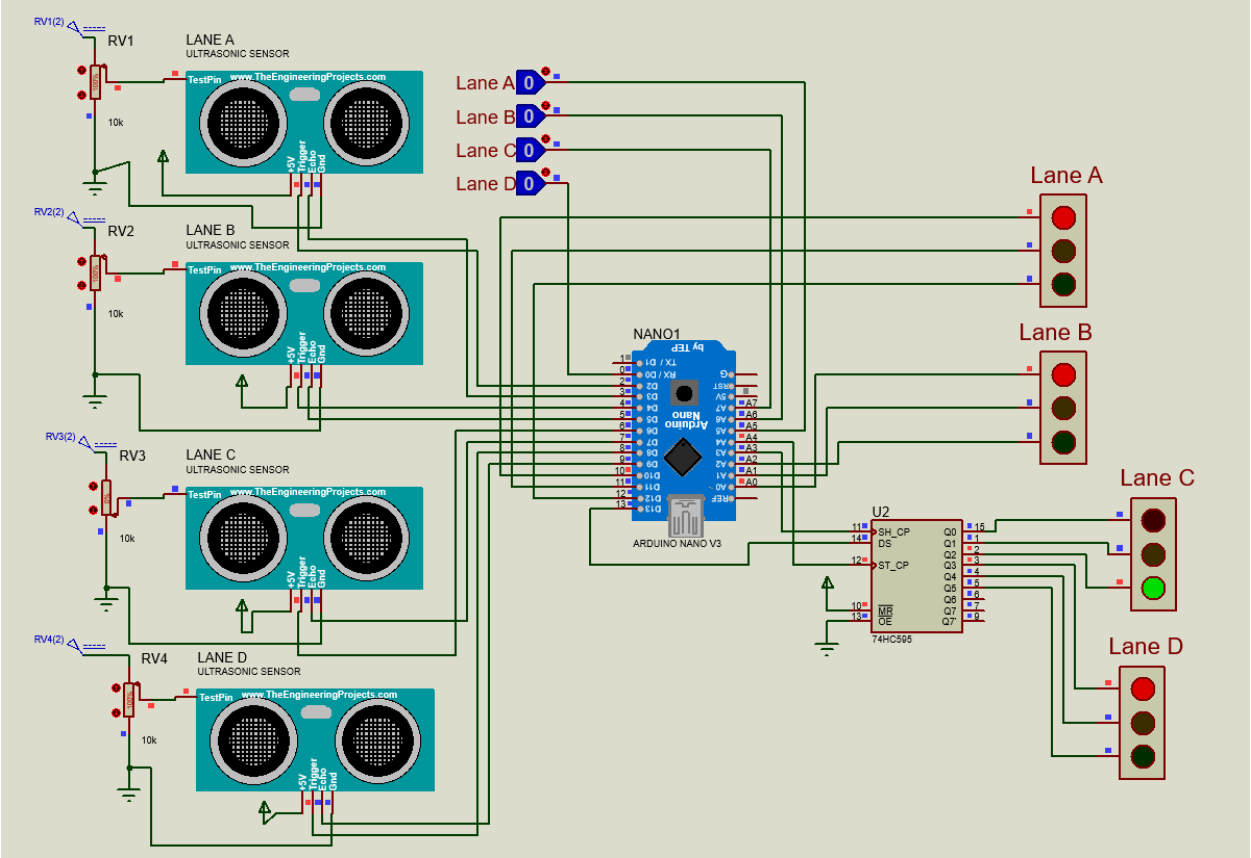

Figure 11: Proteus Simulation

Also, in the simulator, a push-button was provided to simulate the triggering of an emergency vehicle. Real RF based vehicle transmitter was not available in Proteus so for this reason push button was connected to simulate the emergency signal input. If activated, it simulates the receipt of a distressing radio signal and it compels the system to give priority to the lane.

The physical hardware implementation includes a 27 MHz RF transmitter in the emergency vehicle unit and an RF receiver that is connected to the Arduino controller used for the emergency vehicle detection. If the transmitter is turned on, the receiver will recognize the signal and immediately give green light priority to the lane.

Table 6: Technical Comparison between Simulation vs Hardware

Feature	**Simulation (Proteus)**	**Hardware Implementation**
Vehicle Detection	Ultrasonic Sensors	IR Sensors
Emergency Trigger	Push Button	27 MHz RF Transmitter
Signal Processing	Arduino Logic	Arduino Logic
Traffic Control	Virtual LEDs	Physical Traffic LEDs

The logical control algorithm is the same for both simulation and hardware, but the type of sensor used is different. In both implementations:

- Digital signals are created by the presence of any vehicle.
- These inputs are processed by the Arduino.
- Congestion estimation is calculated.
- Traffic signals are turned on or off as appropriate.
- The selected lane is given priority when using emergency override logic.

The simulation phase was important in order to validate the design prior to implementation in hardware. It minimized the chance of development risk, the occurrence of logic mistakes, the correctness of switching timing and the correctness of congestion and emergency handling algorithms. This helped to validate the system in a virtual environment, thereby minimizing hardware debugging time and improving system reliability.

This organised simulation-to-hardware transfer is an instance of the correct engineering development methodology and adds to the general credibility of the hybrid system design.

Hardware Implementation:

The hardware implementation of the proposed hybrid traffic management system is a physical realization of the embedded control architecture which had been designed during the design phase. This system combines congestion based Adaptive Traffic Control (ATC), Emergency Vehicle Prioritization using Radio Frequency (RF) communication and Wrong Way Vehicle (WWV) detection in a distributed dual-microcontroller system. As illustrated in the assembled intersection model, the physical prototype has been built to show the real-time signal control under various operating conditions.

The implementation consists of two Arduino based control units, the main controller, which is in charge of the traffic control and emergency interventions; and the secondary controller, which is only in charge of wrong-way vehicle detection. This separation of architecture helps to isolate tasks, increases the reliability of the system's processing operations, and increases the safety of the system by preventing interference between the optimization logic and violation monitoring.

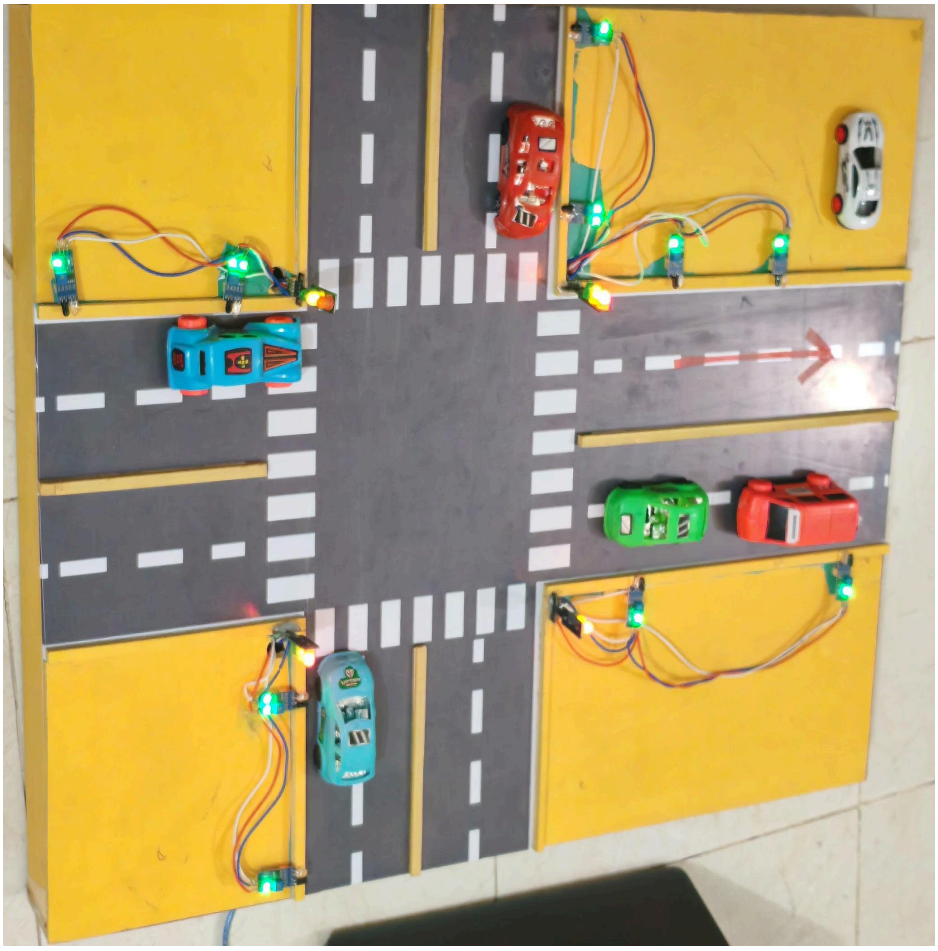

Figure 12: Hardware Setup

Secondary Arduino is used as a Directional Integrity Monitoring unit. Two IR sensors are set up in succession on the same lane so that there is a sequence of directionality in the detection. It is not just about detecting obstacles; the internal mechanism assesses when the sensors are activated. During normal vehicle movement, the first sensor would be triggered before the second sensor. This sequence is continuously tracked by the embedded algorithm. When the order of activation is reversed (ie: the second sensor was activated before the first), this is interpreted as a wrong-way movement event. When this abnormal sequence occurs, the microcontroller activates a buzzer module which is connected to the output pin of microcontroller. The violation is signaled by an audible warning from the buzzer. This mechanism is analytically much stronger than single-sensor detection as it confirms flow direction and not just presence.

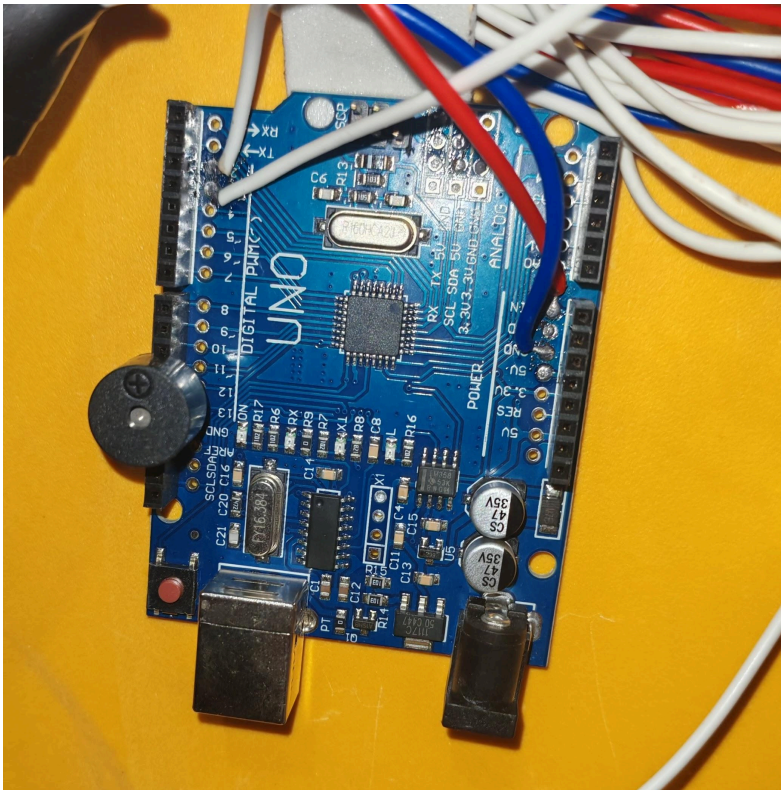

Figure 13: Secondary Arduino

The main Arduino is the central traffic regulation controller. Each lane has two IR sensors and there are eight IR sensors in total. Each lane has these sensors in varying locations that are used to estimate vehicle density. When these digital signals are interrupted by the vehicles the digital signals are sent to the input pins of the microcontroller. The number of sensors in each lane is continually monitored by the controller to gain an idea of congestion levels. The internal logic evaluates the activation pattern of the sensors on the different lanes, and assigns the green signal priority dynamically to the lane with the higher activation density. This adaptive mechanism is used to replace the fixed time signaling with real time sign on based on density optimization, thus making the prototype model more efficient in terms of traffic.

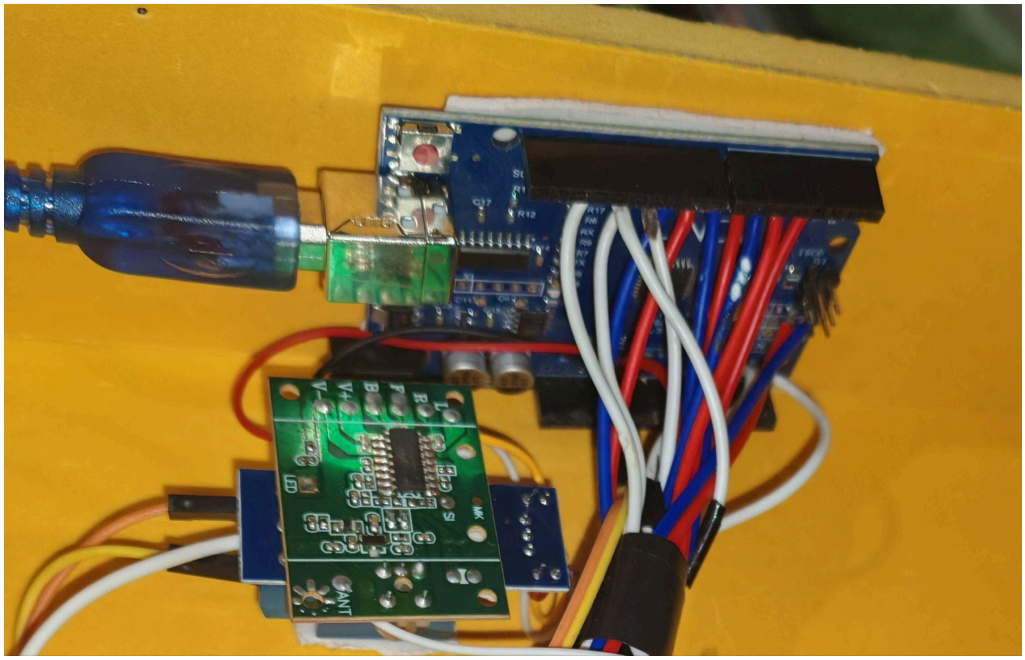

Figure 14: Primary Arduino

The digital output pins of the main Arduino are used for traffic light control. Red, yellow and green LEDs are in each lane to indicate the state of the signal. The controller provides mutual exclusivity so that in normal operation, it is only one lane that receives a green signal while the rest of the lanes receive a red signal. The embedded code controls timing for transitions, ensuring safe switching time.

The wireless RF communication mechanism is used for emergency vehicle prioritization. A 27 MHz RF transmitter module is independently powered in the emergency vehicle unit. Once activated, the transmitter sends out a radio frequency signal which is picked up by the RF receiver module in the traffic controller unit. The receiver is connected to the first Arduino via a signal line that is a digital input line. As soon as the microcontroller receives the RF signal, it immediately aborts the normal congestion-based control algorithm and enters emergency override mode.

A relay module is added in between the microcontroller and the traffic LED circuit to enable the hardware-level implementation of emergency priority. If the Arduino detects the RF signal, then it activates the relay for the emergency lane. The relay is a switching device that is switched on by the power supply, and directly connected to the green LED of the assigned lane without using an electrically isolated device. The mechanical switching ensures that the emergency lane is green no matter what previous software state it was in. At the same time the Arduino makes the other three lanes red to ensure safe passage. A relay adds electrical isolation, protects the microcontroller from higher currents and offers fail-safe physical switching.

This integration of these components may be viewed as a layering of a control hierarchy. In normal conditions, the congestion detection logic controls signal allocation. The congestion logic is superseded by emergency override in the event of an emergency signal. Wrong way detection is independent, and it always operates, regardless of the state of the primary controller. This is a distributed mechanism to keep the safety monitoring active even in case of emergency or congestion based switching event.

Physical intersection model proves system practical. The LED indicators react dynamically to the sensor input, emergency triggering means immediate lane prioritisation is activated and directional violations that result in audible alerts are triggered. The wiring scheme provides common ground and regulated 5V to the microcontroller, sensors, RF modules and relays. High current fluctuations are avoided and electrical isolation by relay switching is provided to increase reliability.

Power Supply and Distribution:

The main arduino controller is powered via its USB port with a regulated 5V. That way the voltage to the system is regulated with the onboard circuitry and enough current is available to power the IR sensors, RF receiver module, relay driver, traffic LEDs and other peripheral devices attached to the system.

The primary Arduino's regulated 5V output pin is directly connected to the secondary Arduino which is used for wrong-way vehicle detection. Both microcontroller are operating on a common ground connection to reference the signals and synchronize the operation. The shared power configuration reduces the need of an extra external power source, reduces wiring complexity and maintains equal power levels across both controllers.
In addition, the relay module will insulate the traffic LED load from the microcontroller, which will help to prevent the excessive currents from affecting the stability of the system. This centralized and regulated power design has the benefits of better reliability, less electrical noise, and helps ensure the hybrid traffic management prototype performs consistently.
The hardware implementation validates the feasibility of the combined platform of embedded control systems, wireless communication and sensor-based monitoring from an engineering analysis point of view. The dual-controller architecture provides reliability by splitting optimization and safety. Real-time priority is independent of the visual detection and is provided by the RF-based emergency detection. The switching via relay adds hardware redundancy that is in addition to the software logic. When taken together, these mechanisms represent a fault-tolerant, scalable and powerful embedded solution that is appropriate for intelligent traffic management applications.

Circuit and Connection Description

The hybrid traffic management system's circuit design includes two Arduino controllers, several IR sensors, RF communication modules, a relay driver, traffic signal LEDs, and a buzzer module. The interconnection structure is designed to provide stable signal flow and correct grounding, while keeping current handling safe.

1. Main Arduino – Traffic Control Circuit Connections:

The main Arduino Uno is used to act as the central traffic controller. The four lanes' traffic signal LEDs are connected directly to the digital output pins. A RED and GREEN LED are located in each lane and a YELLOW LED is shared for all lanes.

Table 7: Pin configuration is as shown below

Function	**Arduino Pin**
Lane A Red	D2
Lane A Green	D3
Lane B Red	D4
Lane B Green	D5
Lane C Red	D6
Lane C Green	D7
Lane D Red	D8
Lane D Green	D9
Common Yellow	D10

The LEDs are powered using a current limiting resistor to avoid overcurrent problems. All the LED cathodes are connected to the ground of the Arduino.

IR Sensor Interfacing for Congestion Detection:
Eight IR sensors are placed on four lanes, two on each lane. These sensors are wired to digital and analog input pins, depending on the output type.

Table 8: Pin configuration for sensors

Lane	**Sensor 1**	**Sensor 2**
Lane A	D11	D12
Lane B	D13	A0
Lane C	A1	A2
Lane D	A3	A4

Digital-output IR sensors are used with digital pins and analog sensors are used with analog input pins. All the sensors are 5V and the ground is common with the main Arduino for stable signal referencing.

Emergency Detection – RF Receiver Connection:
The regulated 5V power supply from the Arduino is used to power the RF receiver module. To ensure the stability of the signal, the module ground is connected to Arduino GND. The RF data output is tied to analog pin A5 which is the emergency detection input. Emergency override is activated when the receiver exceeds a predefined voltage signal when the RF transmitter in the emergency vehicle is activated. This link allows the signal from the RF module to be transmitted to the microcontroller without any intermediate signal processing.

Relay Module Connection and Electrical Isolation:
The relay module is attached to one of the digital output pins of the arduino. When emergency mode is activated a HIGH will be applied to the relay control input. The switching terminals of the relay are in series with the power line of the green LED on the emergency lane that has been identified. If the relay is energized it physically switches the circuit and the green signal is ON for that lane, no matter what the previous condition was. The relay module is used to electrically isolate the load – in this case, the LED – from the Arduino low-current control circuit. This isolation allows for safe switching operation and for the microcontroller to not be impacted by voltage spikes.

2. Secondary Arduino – Wrong-Way Detection Circuit:

The secondary Arduino Uno is separate and is used for wrong way vehicle monitoring. The connections are made as follows:

Table 9: Pin configuration for Secondary Arduino

Component	**Arduino Pin**
IR Sensor 1	D2
IR Sensor 2	D3
Buzzer	D8

The output of both the IR sensors is connected to 5V of primary Arduino and connected to common ground. The buzzer is wired to D8 and to ground. The output pin will provide 5V to the buzzer on activation, creating an audible alert to the operator.

3. Power Wiring and Ground Referencing:

The main Arduino is powered via its USB port which receives a regulated 5V power supply. The 5V output pin of the primary Arduino is used to power the secondary Arduino. A single ground cable is used for both controllers to ensure a common ground for both to prevent floating voltage levels and signal mismatch. The regulated 5V power supply is common to all sensors, RF modules and relay driver circuits. To guarantee signal integrity over digital and analog inputs, common grounding is required. The power configuration is centralized, making wiring easier and providing even power levels to all the components for stable operation.

4. Signal Flow Summary:

Digital or analog signals are received from sensor inputs (IR, RF) to respective Arduinos controller. The congestion and emergency inputs are processed by the main controller, which outputs digital control signals to the LEDs and the relay modules. Directional IR signal is monitored independently by the secondary controller and a buzzer control signal is output from the secondary controller.

The wiring structure guarantees:

- Proper input/output isolation
- Stable voltage referencing
- Safe current handling
- Reliable emergency switching
- Independent safety monitoring

This circuit configuration meets the functional needs of the hybrid traffic management system and ensures electrical safety and reliability.

4.3 Identification of the Optimal Design Approach [CO6]

After evaluation and comparative analysis of proposed alternatives, the hybrid architecture was found to be the best solution due to reliability, comparative robustness, continuity of operations, and practicality.

Through the vision-based system, using a custom-trained YOLOv8 model, the accuracy of vehicle detection, congestion estimation, and emergency vehicle identification were high in controlled conditions with favorable factors. A relevance improvement in detection for real traffic scenarios was achieved by using a finely tuned model trained with a self-developed data set. Yet the success of this method was still heavily reliant on out-of-scope external factors like lighting, camera orientation, weather disturbances and hardware stability. A single-point failure risk exists with any interruption in video feed or failure in the processing system, or obstruction in the environment, that directly affects system functionality.

The hybrid design approach removes these constraints by adding a backup IIoT microcontroller subsystem to provide hardware level redundancy. In this scheme, the vision module is the main intelligence layer and the sensor based embedded system is used as fail safe. The camera system quickly hands over to the microcontroller-based congestion detection system based on IR sensors and RF emergency signaling, if the camera system malfunctions, loses frames, or the interference is unreliable. This makes sure that traffic regulation is maintained without interruption.

The main benefit of the hybrid technique is the layered control. The vision module is responsible for adaptive high accuracy traffic optimization and the microcontroller subsystem ensures the functioning continuity regardless of the surrounding visibility. Emergency vehicle prioritization is also supported by RF communication, that is independent of visual detection. In addition, a separate secondary controller controls wrong way vehicle detection, enhancing system safety.

The hybrid architecture entails a few extra hardware components, but this will be justified by the large boost in reliability and fault tolerance. The distributed control structure is more immune to failure, decreases downtime and helps to make the system more applicable in the real world. In real deployments where the environment cannot be controlled, this redundancy is necessary to ensure traffic is still regulated.

The hybrid architecture offers a balanced and robust solution considering factors such as detection performance, system robustness, emergency responsiveness, scalability, and real-world feasibility. Hence, the hybrid design approach was chosen to be the best configuration, in order to obtain a stable operation, more safety enforcement and better fault tolerance than a purely vision-based system.

Table 10: Comparative Evaluation of Proposed Design Approaches

Evaluation Criteria	**Design Approach 1: Vision-Based Only**	**Design Approach 2: Hybrid (Vision + IIoT)**
Affordability	Reduced hardware costs (camera + processor only)	Slightly more expensive because of extra sensors, RF module, relay and second Arduino.
Ease of Use	Software Control; More straightforward wiring.	Moderate level of complexity owing to additional hardware integration.
Flexibility	Limited visibility – camera only	Maximum flexibility; works both visually and not-visually.
Smart Features	High precision congestion calculation and emergency detection based on advanced AI.	Covers AI intelligence and embedded sensor based redundancy
Reliability	Requires camera stability and environmental factors	Hardware level backup and failover makes it highly reliable.
Sustainability	May fail in poor lighting or due to hardware problems	Sustainable operation even in the case of camera failure or disturbance by the weather.
Practical Feasibility	For use in controlled environments.	More suitable for real world deployment in the presence of uncertainty and variability in the environment
Fault Tolerance	Single-point failure risk	Multiple designs decrease the likelihood of failure
Emergency Handling	AI-based detection only	AI detection + RF hardware enforcement via relay

4.4 Performance Evaluation of the Developed Solution [CO7]

The developed hybrid intelligent traffic management system was tested to determine the functional correctness, operation stability, fault tolerance capability and real-time responsiveness. The performance measures included five key measures: Vehicle detection accuracy, Congestion-based signal adaptability, Emergency vehicle prioritization efficiency, Wrong way violation detection reliability, and System continuity under failure conditions.
Both recorded traffic videos and controlled live video inputs were tested with the primary vision module, which was developed with the custom-trained YOLOv8 vision model, under varying density conditions. The self-developed data set has more than 3000 traffic images, and the model has been trained on this data set, which is why it has better contextual accuracy in the vehicle and emergency vehicle detection task than the generic pretrained model. The congestion estimation algorithm was able to convert the detection counts into adaptive signal control decisions, allocating green signals based on the real-time variation in congestion. In a low traffic scenario, unwanted waiting times were reduced and in heavily congested lanes a higher proportion of priority was given.
In order to assess the robustness, failure scenarios of the camera were also introduced by interrupting the video input and simulating the detection loss. The system was able to switch to the redundant IIoT microcontroller subsystem in the event of failure detection, with minimal delay. The functional signal control without any interruption in the LED switching was achieved in congestion estimation using the IR sensor. This failover validation proved successful for the hybrid architecture to overcome the single point dependency of the vision module.

The use of emergency vehicle prioritization (EVP) was evaluated in both modes. In the vision mode, the emergency classification by YOLO model was done and forced override was applied. During the back-up mode, when the RF transmitter was activated, this resulted in a digital input at the Arduino controller, which would turn a lane to green, but keep the other lanes red. The logic conflicts were avoided and signal integrity was maintained with the help of the relay module, which ensured hardware-level enforcement of the emergency priority.

The wrong way detection subsystem was tested separately by inverting the trigger sequence of both the IR sensors. The secondary Arduino was able to determine whether reverse was being selected or not by looking at the temporal sequence, and then automatically begin to sound the buzzer when it was needed. This subsystem is independent of the congestion or emergency operations, so the system could provide a continuous safety monitoring.
For prototype scale real-time traffic control, the system response time was kept within acceptable limits. Signal transitions were smooth, with no sign of oscillation at any point during mode switching, and continuous operation for long periods of testing showed stable performance with no controller resets or voltage fluctuations.

In sum, the performance assessment validates the hybrid design's ability to provide reliable operations, better safety monitoring, and continued traffic control in both normal and failure conditions. The incorporation of the adaptive image processing with the hardware-level redundancy has proven that the solution developed is feasible for future use in intelligent traffic management.

4.5 Conclusion:

Through the optimization and evaluation process, a reliable hybrid traffic management system was developed that can control congestion in real time, prioritize emergency vehicles, and detect wrong way driving. Several design options were evaluated to trade-off detection accuracy, reliability, cost and practicality.

The vision-based method, with a tailored-trained YOLOv8 model, achieved good performance in vehicle detection and adaptive signal control under normal circumstances. However, it was found that it was dependent on camera stability and environmental visibility, which meant that there were potential reliability limitations.

This weakness was addressed by the hybrid architecture that included a backup microcontroller subsystem running IIoT (Internet of Things). In the event of camera failure, the system automatically switches to sensor-based congestion estimation and RF-driven emergency override, minimising the chance of the single point of failure. It also has an independent wrong-way detection system to further improve safety.

Comprising of performance, robustness, and real-world applicability, the hybrid approach was chosen as the optimum approach. By finalizing the system, it can be made more reliable, continuous to operate and practically suitable to deploy the intelligent traffic management system.

Chapter 5: Completion of Final Design and Validation [CO8]

5.1 Introduction:

The result of chapter is the completion, integration and validation of the proposed hybrid intelligent traffic management system. After optimization and selection of the hybrid architecture, the entire system was implemented and the vision-based detection module (YOLOv8) was connected to the IIoT-based microcontroller backup subsystem. The purpose of this chapter is to ensure that the completed solution is capable of operating reliably in real-time and is able to meet the desired objectives of congestion-based traffic control, emergency vehicle prioritization, wrong way detection, and continuity of operations.

In the last phase of development, refinements were added depending on the results of the performance evaluation. Based on the examination of training metrics like precision, recall, and F1-score, the YOLOv8 model was fine-tuned to achieve a balanced and stable detection performance. From the hardware perspective, hardware calibration of IR sensors, hardware traffic signal timing modification, verification of relay switching mechanism, and emergency override response test were performed to improve the stability and responsiveness of the system.
This chapter satisfies the requirement of CO8 – Completion of the Design and Solution by systematic implementation, integration and structured validation testing. It shows that the final system works as intended and also provides the desired functionality while using a logical and structured approach to engineering.

5.2 Completion of Final Design

This section shows the final implementation of the chosen hybrid traffic management architecture, which includes the required changes determined during its performance evaluation. The testing outcomes of the vision-based subsystem and hardware subsystem were analyzed to make refinements for better detection accuracy, signal stability, emergency responsiveness and system stability. After these changes, this system was completely developed and integrated based on the design configuration determined in the final design. The final solution is the optimized hybrid model which uses an AI-powered traffic analysis and hardware-level redundancy for stable and continuous operations.

5.2.1 Adjustment Based on Performance Evaluation [CO6, CO8]

After detailed performance testing at system level, multiple refinements were made to stabilize detection, responsiveness of the signal and overall reliability of operation. These changes were made based on analysis of the metrics produced by YOLO training, confidence curves and hardware performance in real-time testing scenarios.

- **Adjustment of YOLO Confidence Threshold Using F1 Curve:**

According to the F1-Confidence curve, the model was able to obtain the highest F1 at the confidence level of 0.396 which was around 0.8.

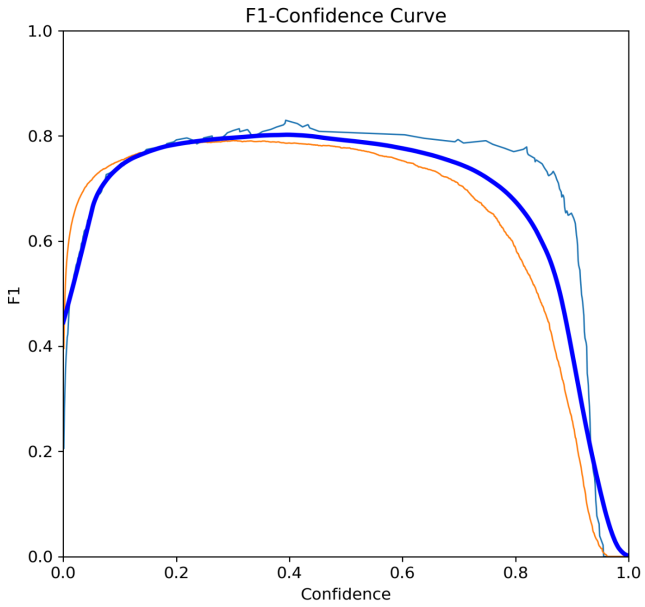

Figure 15: F1-Confidence curve

In the Precision-Recall curve, the overall mAP@0.5 of 0.832 is obtained and the classification results of the Emergency Vehicle class are good.

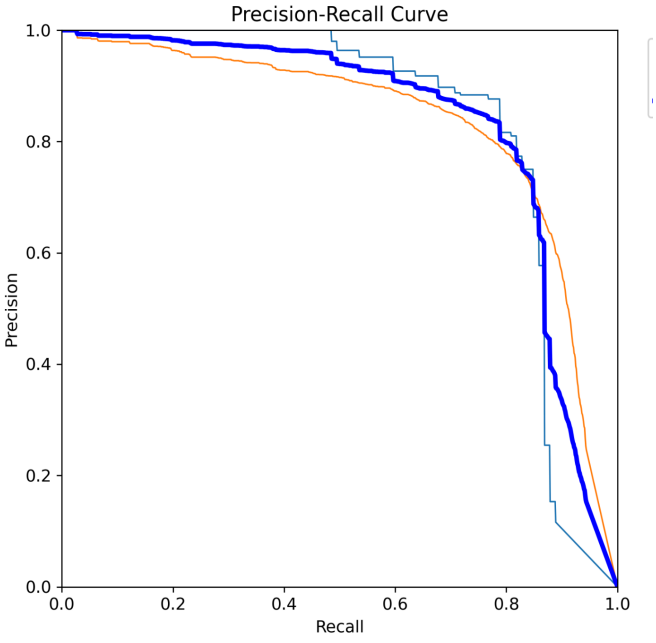

Figure 16: Precision-Recall curve

Furthermore, the Precision-Confidence curve shows a consistent rise in the precision achieved with the rise in confidence, approaching close to 1.0 with the very high confidence values yolo

results. But, too much confidence leads to a serious drop in recall as can be seen in the Recall-Confidence curve.

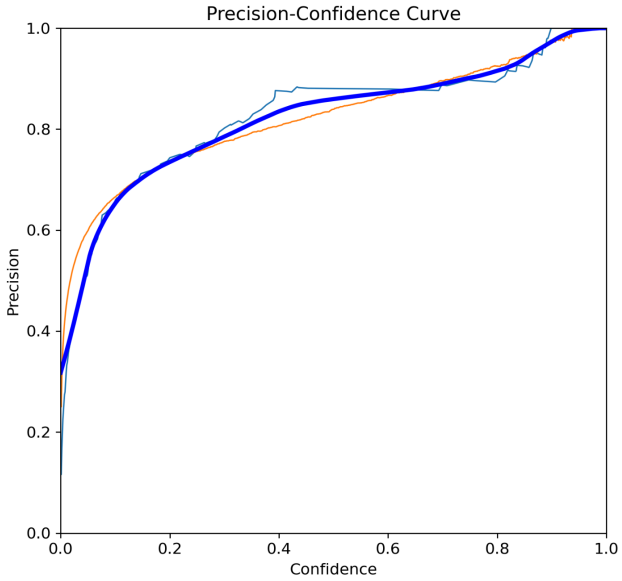

Figure 17: Precision-Confidence curve

From these observations, the deployment confidence level was adjusted to be close to the optimum F1 peak region (approx. 0.4), to ensure:

1. Have correct false positive and false negative levels
2. Secure detection of emergency vehicles
3. Vehicle counting for congestion estimation - Stable.

This tuning has been carried out after the study of the behavior of the models' responses curves in order to avoid responding to any unstable detection behavior during real-time inference.

- **Calibration of IR Sensor Threshold Values:**

Congestion estimation from the hardware subsystems is achieved by using IR sensors that are linked to digital and analog input pins. The analog threshold has been coded in the Arduino as:

```
const int ANALOG_THRESHOLD = 500;
```

In the prototype, this threshold was adjusted during experimental testing to make sure vehicles are detected at different lighting and reflective surface conditions. A low threshold resulted in false triggering of the system and a high threshold resulted in not always detecting vehicles. The chosen threshold produced a stable analog to digital conversion of analog signals but without causing oscillations through iterative testing.

- Adjustment of Traffic Signal Timing:

Signal timing parameters were adjusted based on the observed traffic flow simulation behaviour. The Arduino code specifies:

```
const unsigned long GREEN_TIME_MS  = 4000;
const unsigned long YELLOW_TIME_MS = 2000;
```

Initial testing indicated that less time resulted in more rapid switching and more time resulted in less responsivity. Finalized timing values were able to provide smooth transitions between states, and adaptive congestion prioritization.

- Stabilization of Emergency Override Logic:

Aimed at the emergency detection threshold, the following calibration was performed:

```
if (analogRead(EMERGENCY_PIN) > 800)
```

The threshold was determined through testing to be 800, which prevents activation by electrical noise or signal fluctuation. This made sure that emergency override only occurs if a valid RF signal is received.
In addition, a stabilization delay was added:

```
delay(200);
```

This delay will prevent rapid signal changes and provide stable green enforcement on emergencies.

- **Optimization of Relay Switching Delay:**

To eliminate signal oscillation during emergency activation, relay behavior was evaluated. The programmed short delay, when emergency mode is activated, prevents the relay from switching ON/OFF too many times while keeping the switching stability. This is an optimization at the hardware level that ensures that the green signal is enforced for the lane specified.

5.2.2 Final Integrated System Development [CO7, CO8]

Finally, the entire project was divided into a number of stages for optimizing each of the software and hardware subsystems separately, and then into a single intelligent hybrid traffic control system. The goal for this integration was to make sure the AI based traffic analysis module and the embedded hardware control system could work in tandem. This integration allows the system to integrate smart vision based detection with robust sensor based hardware control to ensure efficient and continuous operation of the traffic signals across various traffic scenarios.
The integration process was executed in a systematic approach and was done by integrating four major subsystems: Vision Based YOLO Detection Module, Congestion Based Arduino Control

System, Wrong Way Detection Subsystem, and RF Based Emergency Vehicle Prioritization Mechanism. The subsystems provide a special portion of traffic management functionality, and they compose together the final operational architecture of the system.

5.2.2.1 Integration of Vision-Based YOLO Detection Module

The vision module is the key component for traffic monitoring and decision support in the hybrid system, which is based on the YOLOv8 algorithm. The trained deep learning model detects the traffic image from the camera and detects the object at different lanes, which includes vehicles, emergency vehicles, etc. The model was trained with a custom-made dataset of over 3000 images with traffic scenes captured from real life situations, increasing the contextual relevance of the detection process.

In real-time operation, the camera takes continuous photos of the traffic frame and uses the YOLOv8 inference engine to process them. The model outputs the bounding boxes and class labels for the vehicles detected. Based on these detections, the system can determine the density of vehicles and detect emergency vehicles, if present. With this information the system can come to understand the current traffic situation, and provide intelligent traffic signal decision support.

The addition of the YOLO module allows the system to dynamically analyse traffic congestion and identify emergency vehicles using computer vision techniques. This software-driven intelligence offers adaptive traffic control capability, but not just based on fixed sensor inputs.

5.2.2.2 Integration of Congestion-Based Arduino Control System

In addition to vision-based monitoring module, an hardware based traffic signal control subsystem was designed using an Arduino microcontroller. This subsystems controls the physical traffic signal LEDs and traffic flow based on the congestion detection.

Several IR sensors are placed on each traffic lane, providing their data to the Arduino controller. In the prototype setup, the two sensors are placed in each lane to indicate the presence of a vehicle at a particular location in the lane, which can be used to determine the traffic density in the lane. The controller uses the signals from the IR sensor beams to detect congestion as cars cross the beams.

This sensor information is fed into a microcontroller which has a built in programmed traffic control algorithm to prioritize the signal for lanes with greater traffic volume. Congested lanes can be given the green light, while other lanes are in red. Signal timing states change from green, yellow to red, and vice versa, using pre-programmed timing parameters.

This congestion-based control system allows for hardware-level sensing of traffic to make traffic signal decisions even when the vision module is not working.

5.2.2.3 Integration of RF-Based Emergency Vehicle Prioritization

An RF based emergency vehicle detection system was incorporated into the system so that the emergency vehicles like ambulances can pass through the system as soon as possible. This

subsystem is comprised of an RF transmitter in the emergency vehicle and an RF receiver to the traffic controller.

As the emergency vehicle approaches an intersection, the RF transmitter emits a signal—this signal is received by the RF receiver module that is connected to the Arduino controller. When the controller receives this signal, it quickly activates the emergency override logic.

Overrides will cause the traffic signal in the emergency vehicle lane to stay green and all other lanes to be red. Such operation is carried out via a signal switching relay module. The relay offers electrical isolation, and ensures that the emergency signal is not affected by the normal traffic control logic.

By incorporating this subsystem, emergency vehicles are given top priority information, which makes emergency response more efficient, thereby ensuring the safety of traffic.

5.2.2.4 Integration of Wrong-Way Detection Subsystem

Besides the traffic flow management facility, the system also includes wrong way vehicle detection to improve road safety. This subsystem is independent and is controlled by another Arduino controller, with a pair of IR sensors placed in the lane one after the other.

The detection mechanism is based on the sequence of the sensor activation. When traffic is normal, vehicles flow through the sensors in a certain order. But when a vehicle is traveling in the opposite direction the order of the sensor activation is reversed. This sequence of sensors is constantly monitored by the microcontroller to determine violations of direction.

If a wrong-way is detected, a buzzer alarm is sounded to notify authorities or monitoring personnel in the immediate vicinity. This alert system can quickly identify violations of traffic laws, and prevent any potential accidents.

This subsystem provides another layer of safety monitoring to the whole traffic management system.

5.2.2.5 Integrated Hybrid System Operation

All the subsystems have been integrated and the whole hybrid traffic management system has been tested in various working scenarios. The integrated framework enables the vision module, congestion sensors, emergency override system, and wrong-way detection unit to be controlled as a coordinated whole.

Normally, the vision-based YOLO module analyzes the traffic and helps to estimate the traffic congestion in real time. At the same time, the hardware controller controls the signal transitions, according to the inputs from sensors. The system automatically switches to the emergency lane when an emergency vehicle signal is received via RF communication. Also, the wrong way detection module will constantly watch for directional violations and trigger alerts as needed.

Experimental testing showed that the integrated system can be validated to perform adaptive traffic signal control, emergency vehicle prioritization, and wrong-way detection and ensure the stability of signal transitions. AI detection and hardware control provide enhanced reliability and continuity of operation.

The systematic and logical completion of the project design (PO (c6)) is reflected in the successful integration of these components. The final system architecture is thus a feasible hybrid approach with both intelligent vision processing and powerful embedded control to enable efficient and reliable traffic handling.

5.3 Evaluation of the Solution to Meet the Desired Need [CO6, CO8]

This section tests the performance of the completed hybrid traffic management system to assess whether the solution developed meets the projects goals. The software-based vision module and hardware-based embedded control subsystem was evaluated by experimental testing. The system was tested on various scenarios such as congestion detection, prioritisation of emergency vehicles, detection of wrong way vehicles and failover operation in hybrid mode.
Main objectives of the evaluation are: to check whether the solution that has been developed is able to manage the traffic flow, improve efficiency of the emergency response service, detect violations of traffic regulations, and ensure uninterrupted service in case of failure. These validation experiments will guarantee that the system developed will be in accordance with the project requirements and PO(c4) that involves assessing if the design solution is what is needed.

5.3.1 Congestion-Based Traffic Optimization [CO6]

The system's congestion management features were assessed by analyzing traffic signal performance in various traffic density scenarios. Vehicle density detection was carried out using two different approaches – a vision-based detection module using the YOLOv8 object detection model and an IR sensor-based congestion detection system that was integrated with the Arduino controller.
The trained YOLOv8 model showed a high overall mean Average Precision (mAP) of about 0.832 at an intersection of 0.5, demonstrating its ability to accurately detect vehicles and emergency vehicles. The F1-confidence curve demonstrated the best detection performance around a confidence threshold set at around 0.396, which was chosen for real-time deployment to achieve a good balance between precision and recall.
The system was tested in experimental conditions and it was possible to identify the lanes that contained more vehicles and dynamically assign the green signal to these lanes. The control logic, implemented on the Arduino, received the sensors data measured for each lane and took the decision about the traffic priority depending on the congestion level measured by the sensors. The state and the timing parameters of the transition between the green, yellow and red states were stable and met the pre-defined parameters.
Results indicate that the system is able to optimise traffic flow by assigning signal priority according to traffic conditions, which can minimise the waiting time for the more congested lanes.

5.3.2 Emergency Vehicle Prioritization [CO7]

The RF transmitter–receiver communication mechanism inside the traffic controller was used to evaluate emergency vehicle prioritization. As soon as the emergency transmitter was activated, the RF receiver attached to the Arduino controller picked it up and activated the logic for emergency mode instantaneously.

When the emergency signal was received the traffic controller switched the corresponding lane signal to "green" and switched all the other lane signals to "red". This operation was made possible with the aid of the relay module, and the signal switching was made reliable at hardware level without the interference of the normal traffic control algorithm.

Test results showed the emergency override system worked very quickly and reliably and that emergency vehicles were able to pass through the intersection without delay. This feature can greatly facilitate emergency response and generate a higher safety level for the roads.

5.3.3 Wrong-Way Detection [CO7]

Sequential sensor triggering tests were used to evaluate the wrong-way vehicle detection subsystem. Two IR sensors are used in the system to feed into a secondary Arduino controller which tracks the direction of vehicle movement.

Normally, cars go past the sensors in a specific order when there is no congestion. The sequence of sensors that trigger, however, is the reverse when a vehicle is traveling in the opposite direction. This sequence is monitored by the microcontroller and any violations of direction detected will be noted.

When the sensor activation order was reversed for testing, the system was able to tell when the vehicle was moving the wrong way, thus sounding the buzzer. The instant notification system enables officials to pinpoint traffic offenses and implement corrective measures.

The experimental results show that the wrong way detection subsystem works well and becomes another safety monitoring function of the whole traffic management system.

5.3.4 Hybrid Failover Capability [CO6]

The other important goals of the proposed system is to avoid the single point failure through a hybrid architecture. The failover ability of the system was tested by simulating loss of the camera-based vision module.

In these tests the system itself used the hardware congestion detection subsystem automatically, which is based on IR sensor inputs. No traffic signal operation was interrupted while the Arduino controller continued to control the traffic signals based on the data from the sensors.

This hybrid way of operation guarantees that traffic control functionality continues to operate even if the camera system fails due to hardware malfunction or environmental issues like poor lighting. The operation of the backup subsystem is successful, which improves the reliability and fault tolerance of the system.

5.3.5 Performance Summary [CO6, CO8]

To summarize the overall performance of the developed system, an evaluation was done for each major functionality based on the desired project objectives.

Table 11: System Function and Evaluation Result

System Function	Evaluation Result	Observation
Vehicle Detection (YOLOv8)	Successful	mAP@0.5 is close to 0.83, providing a reliable vehicle detection.
Congestion-Based Signal Control	Successful	Dynamically allocated green signal for congested lanes
Emergency Vehicle Prioritization	Successful	When an override on the RF side occurs, the green signal is activated at once.
Wrong-Way Detection	Successful	Buzzer alert is triggered correctly by the sequence of Sensors
Hybrid Failover Operation	Successful	In camera failure, the hardware subsystem takes over traffic control.

The results of the evaluation show that the proposed hybrid traffic management system has fulfilled the desired functional requirements and possesses application feasibility to intelligent traffic control.

5.4 Conclusion

In this chapter, the completion and validation of the proposed hybrid intelligent traffic management system has been presented. Finally, the integrated vision detection module based on YOLOv8 and the embedded hardware control subsystem based on IIoT were successfully implemented. In the last development phase several refinements have been added to the algorithm, based on the performance evaluation results, in order to enhance the detection accuracy, the stability of the signals and the reliability of the system.

The evaluation findings proved that the system developed meets the main goals of the project. The trained YOLOv8 model effectively detected vehicles and emergency vehicles, allowing for real-time traffic monitoring and congestion analysis. The congestion detection module, implemented on the Arduino microcontroller, was able to effectively control traffic lights based on the input data from the IR sensors, dynamically allocating green light to lanes with more traffic.

The validation of emergency vehicle prioritization has been done on the RF transmitter-receiver system, which will make the signal green in emergency lane and not in other lanes with the help of relay based override. Further, the wrong way detection sub-system was able to detect wrong way using sequential triggering of the sensors and an audible warning was given by buzzer module.
Moreover, the hybrid system allowed traffic signals to operate normally even if the camera module was down, since the sensor-based hardware subsystem continued to control traffic signals. In conclusion, the implementation and testing have validated the system's functionality and its potential feasibility for intelligent traffic management applications.

Chapter 6: Impact Analysis and Project Sustainability [CO3, CO4]

6.1 Introduction

Solutions in engineering systems for use in the real world should be considered for their impact on society and the environment. The focus of this chapter is the analysis of the effect of the proposed hybrid intelligent traffic management system from different perspectives, such as from a societal point of view, safety, legal and cultural aspects, and sustainability. The goal of this chapter is evaluate the developed solution and its impact in the improvement of urban traffic conditions, taking into account some responsible engineering practices.
The system involves combining a vision module using YOLOv8 with an embedded microcontroller subsystem for the management of traffic congestion, prioritization of emergency vehicles, and wrong-way traffic detection. The system is designed to use AI to optimize traffic flow and enhance road safety through the use of sensors and hardware control. But the introduction of such technology cannot be without its influences on society, public safety, regulatory requirements and environmental issues.
Accordingly, this chapter discusses the effects of the proposed solution on society and safety, considers transportation impacts and discusses the environmental sustainability of the design. The project is also mapped to relevant United Nations Sustainable Development Goals (SDGs):

- SDG 9 (Industry, Innovation and Infrastructure)
- SDG 11 (Sustainable Cities and Communities)
- SDG 13 (Climate Action)

The chapter illustrates with this assessment how the proposed intelligent traffic management system helps in the development of safer and more efficient and environmentally friendly urban transportation infrastructure.

6.2 Assess the Impact of the Solution [CO3]

An intelligent traffic management system can have a major impact on contemporary transportation infrastructure. The proposed hybrid system can solve various urban problems

including traffic congestion, emergency vehicles delay and road safety violations. The following subsections analyse the society, health, safety, legal and cultural impacts of the proposed solution.

6.2.1 Societal Impact

The city traffic is one of the main concerns in many cities yet to be developed and it involves a lot of delays, fuel wastage and economic losses. The idea of intelligent traffic management system is to alleviate traffic congestion by dynamically assigning traffic lights according to the current traffic density and vehicle detection. The system prioritizes lanes with higher levels of congestion, helping to improve traffic flow and minimize the need for unnecessary waiting times at intersections.

Further, there is a significant improvement in the efficiency of emergency response with the system. At busy intersections, ambulances and other emergency vehicles can be held up, potentially impacting medical response times. The emergency override via radio frequency has been put in place so that emergency vehicles always have priority of the signal and can pass without delay.

The system is also well suited for the development of smart city infrastructure, in which intelligent transportation systems are a major element to enhance urban mobility. The proposed solution includes AI-based traffic analysis and embedded sensor networks, which can help to optimize transportation management and enhance technological advancements.

6.2.2 Health and Safety Impact

Road traffic crashes and other traffic safety infractions are significant public health problems. The proposed system has multiple ways of improving the safety of the roads. One of the initial advantages of congestion-based traffic control is that it minimizes random traffic flow by the way of more seamless signal changes and more effective traffic management. This helps to reduce the risk of intersection collisions.

Second, there is a wrong way vehicle detection (WWVD) subsystem which detects vehicles traveling against the direction of traffic flow. A leading cause of serious road crashes is wrong way driving. The detection mechanism is a sequential IR sensor logic to detect the directional violations and give a buzzer alarm so that the authorities can take corrective measures immediately.

Third, the emergency vehicle prioritization feature indirectly fosters positive public health results by allowing ambulances to move quicker through traffic intersections. In medical emergencies, time plays an important role in the process of emergency response and the proposed system reduces delay in emergency services.

6.2.3 Legal and Regulatory Considerations

Traffic Management Systems shall be operated within the framework of laws and regulations. The proposed solution can be implemented in a way that will comply with current traffic

regulations, with normal operation of the signals and the prioritization of emergency vehicles following the normal traffic management practices.
The use of emergency vehicle prioritization is already known in several countries and the use of the override mechanism based on RF fits this requirement. The system allows emergency vehicles to get through intersections first, without compromising traffic safety.
Moreover, the system design can be modified to meet the requirements of local Transportation Authorities and the local smart city rules. Since the proposed system preserves the normal traffic signal functions and keeps the traffic signal safe to operate, it can be easily introduced into the current traffic network without conflicts with the regulations.

6.2.4 Cultural and Social Acceptance

For the effective use of intelligent transportation technologies, they have to be accepted by the public and match local traffic behaviour. Many drivers in many cities often get frustrated at the inefficiency of traffic signals and unpredictable congestion. A system which adapts signal based on actual traffic conditions is likely to be well accepted by travellers on the road.
Also, automatic identification of traffic violations, including wrong way driving, can help promote good driving habits and traffic safety awareness. With the growing adoption of intelligent traffic systems in cities, it is expected that the public will accept such systems.
The proposed solution contributes to improved and more organized traffic behavior and the advancement of technologies in the management of traffic.

6.3 Evaluate the Sustainability of the Solution [CO4]

Consideration of sustainability in engineering design today is important. The design of the proposed hybrid intelligent traffic management system was designed with sustainability, affordability and energy efficiency in mind. The integration of a vision-based traffic detection module and low-power embedded hardware enables the system to be both effective and resource-efficient for controlling traffic in urban areas.
The proposed system is one of the advantageous features due to its low cost of implementation. The developed prototype costs approximately 19,387 BDT (without considering the cost of the processing computer used for the YOLO model). The relatively low cost makes the system economically viable for deployment in developing regions where deployment of large scale intelligent transportation infrastructure may be financially difficult. Moreover, the system is cost-effective and easy to maintain, thanks to the availability of a range of components like Arduino microcontrollers, IR sensors, and RF modules.
In addition, the system is designed to be power efficient, enabling it to be used for long term continuous operation. Optimized software processing and energy efficient hardware components enable the solution to operate without the need for high energy resources. These features make it a feasible and sustainable solution for today's traffic management applications.

6.3.1 Environmental Impact

When traffic jams occur, the vehicles spend more time idle waiting at intersections, which results in greater fuel consumption and greenhouse gas emissions. The proposed system ensures optimization of vehicles using the congestion based signal control, which reduces unnecessary waiting time of vehicles. Dynamically adapted traffic signals allow for vehicles to idle less, which leads to a reduction in pollutants including carbon dioxide (CO_2) emissions and nitrogen oxides, and lower fuel usage.

Better traffic also leads to better urban air quality and helps reduce environmental pollution from road transport. Thus, the proposed system can be used for environmentally responsible traffic management.

6.3.2 Energy Efficiency of the System

The hardware components used in the proposed system has low power consumption hence the system is energy efficient. Unlike traditional large-scale traffic monitoring systems, microcontrollers like the ARDUINO boards use very little electricity, as do the IR sensors and RF modules.

The fact that low voltage is needed for the operation of the system means it may have the potential to be powered through other renewable energy sources that are often integrated into traffic signal infrastructure, like solar panels. This feature contributes to the system's sustainability by minimizing reliance on traditional energy sources and allowing for environmentally friendly installation in remote or off-grid locations.

6.3.3 Long-Term Sustainability and Scalability

Proposed hybrid traffic management system has a modular architecture with embedded hardware control and AI based software. The modular architecture enables upgrading or expansion of the system without having to replace the entire infrastructure. Further sensors, cameras or communication modules can be added as traffic monitoring needs change.

Moreover, the low cost and scalability make the system applicable to the developing countries where intelligent transportation systems are just beginning to develop. The system can be expanded to become a part of the centralized traffic monitoring platform or smart city infrastructure, thus enabling large-scale traffic optimization.

In conclusion, the proposed solution is sustainable and practical, with its low cost, low power consumption, modular design, and scalability making it an attractive choice for intelligent traffic management.

6.3.4 Alignment with United Nations Sustainable Development Goals (SDGs)

The envisioned intelligent traffic management system makes several United Nations Sustainable Development Goals (SDGs) possible.

- **SDG 9** – Industry, Innovation and Infrastructure: The project encourages innovative infrastructure development by introducing AI and embedded systems in transportation management.

- SDG 11 – Sustainable Cities and Communities: Overall, the system helps enhance traffic efficiency and emergency response capabilities, leading to safer and more sustainable urban transportation.
- SDG 13 – Climate Action: Better traffic flow and reduced vehicle idle time helps reduce carbon emissions, which helps support the environment.

These contributions show that the proposed engineering solution is in line with the global efforts towards sustainable and resilient urban development.

6.4 Conclusion

The impacts of the proposed hybrid intelligent traffic management system to the society, safety, law, culture and environment were addressed in this chapter. The analysis shows that the system has the potential to positively affect urban transportation in terms of better traffic efficiency, emergency vehicles reaction, and better road safety.

Societally, the system can alleviate congestion and facilitate the smart transportation infrastructures. Road safety and public health outcomes are enhanced by the safety features that have been put in place, such as emergency vehicle prioritization and wrong-way detection.

The environmental analysis shows that optimised traffic flow can lead to a reduction in vehicle idle time, lower fuel consumption, reduced emissions and improved air quality. Moreover, the power-efficient hardware implementation and modular design of the system ensure its sustainability and scalability.

The overall solution has a positive societal and environmental impact and is in line with the global goal of sustainability, e.g., United Nations Sustainable Development Goals. The analysis indicates that the designed system is a responsible and sustainable engineering solution for the current smart traffic management.

Chapter 7 : Engineering Project Management

7.1 Introduction :

Engineering project management is the systematic process of planning, organizing, executing and controlling engineering projects to achieve objectives within constraints of time, cost and quality. It's more about managing the technical aspects, resources and team operations and getting them to work together in a seamless manner, ensuring that projects are delivered successfully without cost or time overruns. For projects such as the smart traffic management system, where hardware, software, and communication technologies are integrated, good management practices are crucial for ensuring seamless integration, risk management, and project success. It also helps in the standardization aspects, improves the stakeholder

communication and eventually facilitates the migration of technical ideas to reliable, scalable and working ones.

7.2 Define, plan and manage engineering project :

7.2.2 Project Plan (EEE499 P) :

Table 12: Project Plan (FYDP P)

Tasks	**Start Date**	**End Date**	**Duration**
Topic Selection	11 Feb		
Initial brainstorming	11 Feb	15 Feb	5 days
Narrowing down options	14 Feb	20 Feb	7 days
Final selection	17 Feb	22 Feb	6 days
Literature Review	18 Feb	20 Mar	31 days
Gathering sources	18 Feb	28 Feb	11 days
Initial reading	22 Feb	10 Mar	17 days
Synthesis and notes	1 Mar	20 Mar	21 days
Data collection	15 Mar	5 Apr	22 days
Analysis	25 Mar	10 Apr	17 days
Content drafting	1 Apr	25 Apr	25 days
Review and edits	15 Apr	5 May	21 days
Initial sketches	26 Feb	8 Mar	11 days
Designing Multiple Approaches	3 Mar	12 Mar	10 days
Refinement	8 Mar	18 Mar	11 days
Identifying risks	12 Mar	20 Mar	9 days
Evaluation	10 Mar	18 Mar	9 days

Ethical considerations	13 Mar	20 Mar	8 days
Proposal Preparation	25 Apr	7 May	13 days

7.2.3 Gantt Chart (EEE499 P):

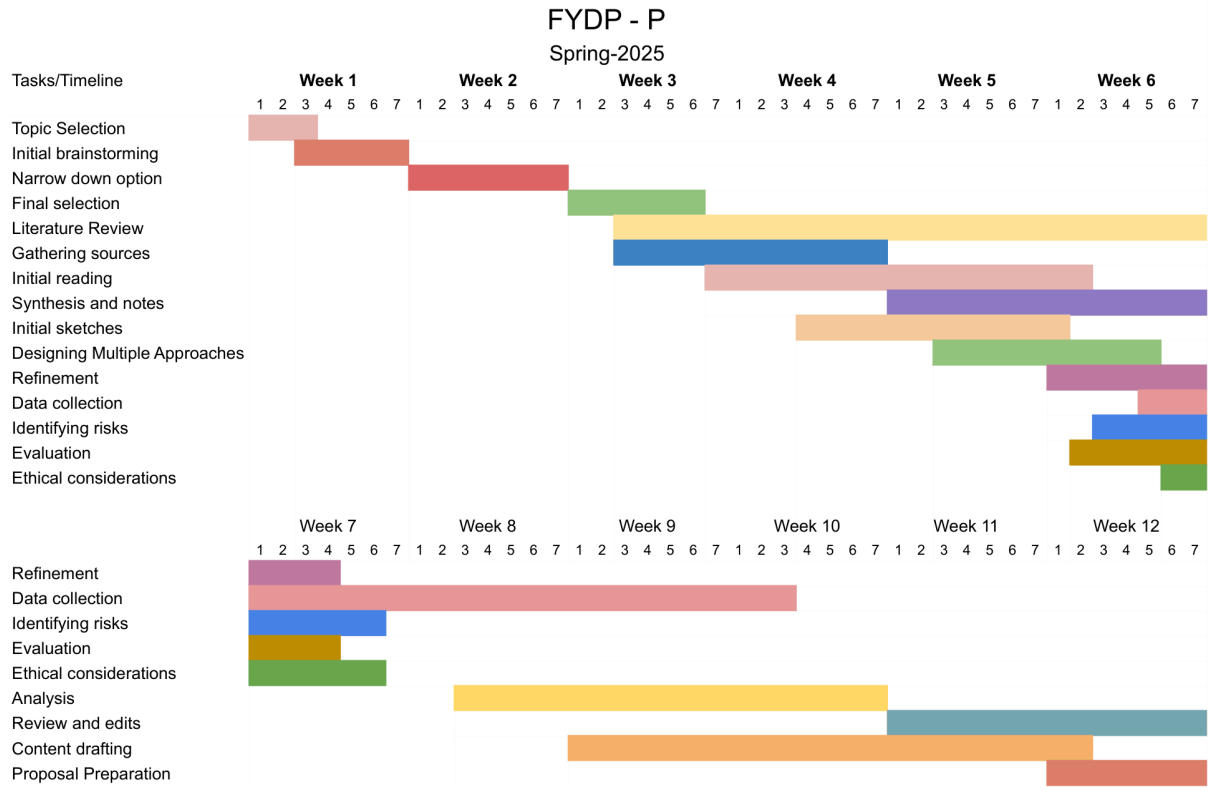

Figure 18: Gantt Chart FYDP-P

7.2.4 Project Plan (EEE499 D) :

Table 13: Project Plan (FYDP D)

Task Name	Start Date	End Date	Duration
Refine Design Plans	17 Jun	30 Jun	14 days
Initial Review	17 Jun	25 Jun	9 days
Adjustments	24 Jun	5 Jul	12 days
Finalization	3 Jul	10 Jul	8 days
Design 1 Simulation	8 Jul	20 Jul	13 days
Running Tests	15 Jul	25 Jul	11 days
Design 2 Simulation	22 Jul	5 Aug	15 days
Running Tests (2)	28 Jul	10 Aug	14 days
Result Analysis	1 Aug	15 Aug	15 days
Results Refining	10 Aug	20 Aug	11 days
Comparative Analysis	12 Aug	22 Aug	11 days
Selection of Approach	20 Aug	25 Aug	6 days
Documentation	10 Aug	5 Sep	27 days
Final Presentation and Report	25 Aug	27 Aug	3 days
Report Drafting	28 Aug	10 Sep	14 days

7.2.5 Gantt Chart (EEE499 D):

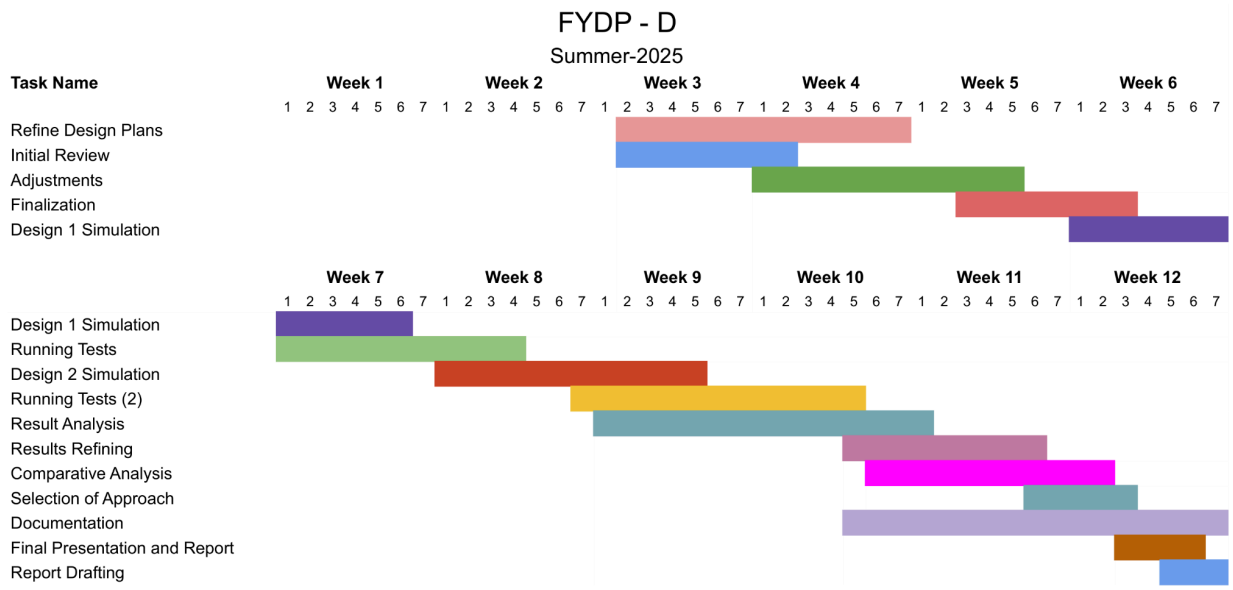

Figure 19: Gantt Chart FYDP-D

7.2.6 Project Plan (EEE499 C) :

Table 14: Project Plan (FYDP D)

Task Name	Start Date	End Date	Duration
Initial Meeting	7 Oct	10 Oct	4 days
Review Options	8 Oct	15 Oct	8 days
Feedback Collection	12 Oct	20 Oct	9 days
Source Collection	15 Oct	25 Oct	11 days
Outline Drafting	20 Oct	30 Oct	11 days
Real Data Collection	25 Oct	20 Nov	27 days
Components Finalization	5 Nov	25 Nov	21 days
Test 1	20 Nov	30 Nov	11 days
SWOT Analysis	25 Nov	5 Dec	11 days
Test 2	1 Dec	10 Dec	10 days
Budget Planning	5 Dec	12 Dec	8 days
Final Budget	10 Dec	15 Dec	6 days
Safety Planning	10 Dec	18 Dec	9 days
Ethical and Safety Considerations	12 Dec	20 Dec	9 days
Report Outline	15 Dec	25 Dec	11 days
Slide Preparation	20 Dec	4 Jan	16 days
Final Edits	26 Dec	4 Jan	10 days
Final Rehearsal	28 Dec	3 Jan	7 days

7.2.7 Gantt Chart (EEE499 C) :

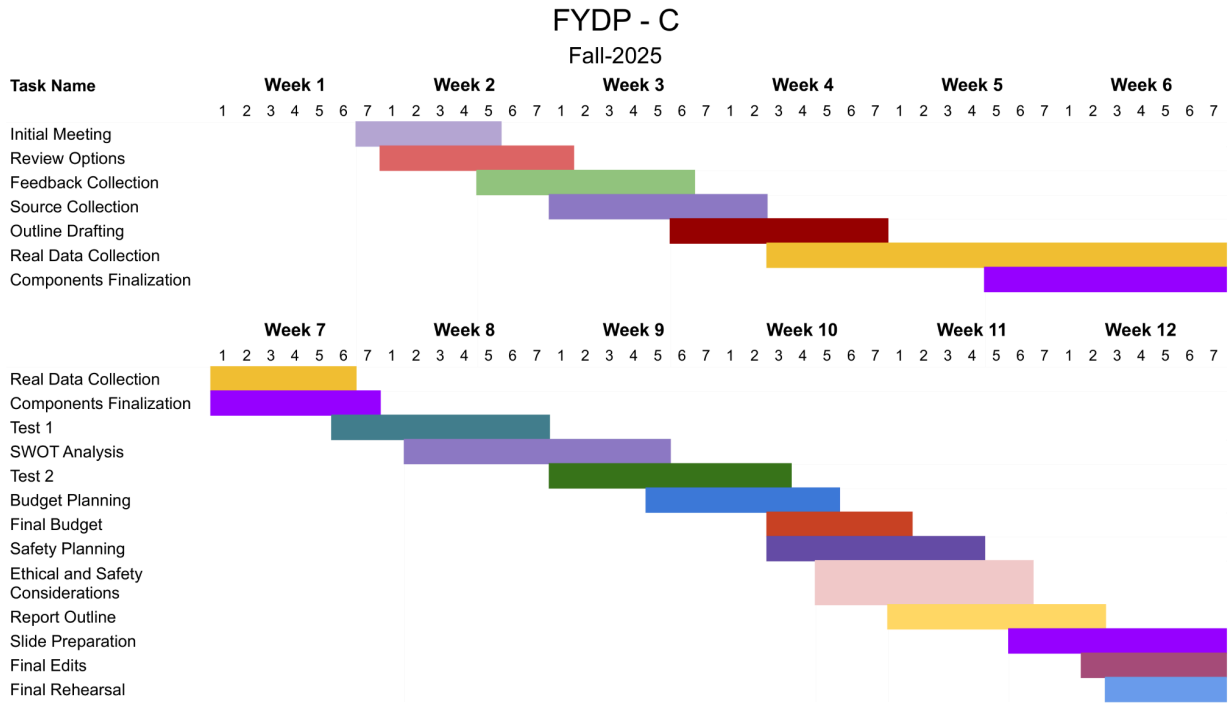

Figure 20: Gantt Chart FYDP-C

7.3 Evaluate Project Progress :

The Smart Traffic Management System underwent a series of rigorous tests at different stages of development to ensure its reliability, responsiveness, and operational efficiency in real-time traffic scenarios. Initially individual components were tested, namely the Raspberry Pi, the USB camera module, the Arduino Uno microcontroller, the IR sensors, the RF transmitter/receiver units, the LEDs and the buzzer unit. In order to evaluate the camera module, tests were conducted under various lighting conditions to check the detection accuracy of vehicles and emergency vehicles based on the YOLOv8 model. The same applies to the IR sensors, which were tested for accurate triggering at various distances and angles to ensure that the vehicle density measurement and wrong-way detection logic is correct.

Testing the RF communication system involved sending the signals from the prototype emergency vehicle and receiving the signals at the receiver module to ensure that they are received in time and in the right order. The control system using the Arduino was investigated ensuring the proper synchronization among the input of the sensors and the actions carried out by the system, such as switching the traffic signals and activating the buzzer. The dual IR sensor setup (Sensor A and Sensor B) for wrong-way detection was tested with several different vehicle directional movements and the results validated that they would only alarm if a vehicle was travelling in reverse.

Integration testing was done to ensure that the primary AI based system and backup IoT based system are able to coordinate seamlessly. The system was tested for fail-safe operations in which the camera input was intentionally tampered with and the system successfully switched to the Arduino based control. YOLOv8 model was tested with a carefully curated set of traffic images and live video feeds, demonstrating its accuracy in detecting vehicles and its ability to accurately identify emergency vehicles.

A small-scale prototype intersection model was used to conduct the full system test, under real-world simulated traffic conditions with various vehicle densities and emergency scenarios. The system was tested for the time of response, accuracy of switching signals and priority handling. The results indicated that the system was efficient to allocate green signal to the most congested lane maintaining immediate priority to the emergency vehicle. With multiple detections of emergencies, the system correctly prioritized based on the order of detection. Performance optimization was conducted to make the system more responsive, and to minimize delay in the signal transition and increase the accuracy of the decision. Through the evaluation, it was found that the system is robust, adaptable and can run steadily in various simulated situations, so as to realize the intelligent traffic management.

7.4 Risk Management Matrix :

Table 15: Risk Management

Risk Event	**Potential Impact**	**Management Procedure**	**Contingency Plan**
Failure of camera or inadequate visibility (fog, light conditions, obstruction).	Wrong traffic decision, YOLO is not able to detect individual vehicles.	Apply controlled lighting set up in prototype, make sure the camera is calibrated, make routine checks of the system.	Implement backup IR + RF based system for traffic control
Misclassification of YOLO model (vehicle vs emergency vehicle)	Wrong lane prioritisation traffic delay/emergency misrouting.	Expand the training of the dataset by adding images of a variety of emergency vehicles; test for model validation.	If RF-based emergency detection is detected, it overrules YOLO decision.
Raspberry Pi Processing Delay/Overload.	Slow traffic response, lagged signals	Optimize code, reduce frame rate, use lightweight YOLO version.	Change Control to Arduino backup system
Failure of IR sensor(s) or triggering false alarms.	Incorrect congestion estimation	Multiple IR sensors per lane (sensor redundancy); calibration prior to testing.	Using a camera and YOLO system for reliable operation
Failure of RF communication (emergency vehicle signal not received)	Emergency vehicles not prioritized: 106.7% of the total.	Perform strong RF range testing and frequency tuning	Emergency detected using YOLO camera system.
Arduino Uno malfunction	Loss of backup system and failure of wrong way detection.	Testing of hardware prior to integration; watchdog reset logic.	Primary Raspberry Pi system continues operation

Misalignment of the wrong way detection sensor (IR A/B error)	The occurrence of false alarm or failure to detect a violation.	To achieve careful physical alignment and lane calibration	Manual override and/or system reset for testing
Failure of prototype system due to power loss.	Complete system shutdown	Relay on a dual power supply (USB + external battery backup device).	Safe default mode: all red signal / manual control
Failure of Traffic LED signal.	Wrong traffic control signage leading to confusion	Test LED circuits (pre-test); add current limiting resistors	Take over the manual control mode.
There is a data synchronization problem between Pi and Arduino	Conflicting decisions (between systems)	Establish priority rule: Raspberry Pi > Arduino (except backup emergency mode)	Arduino with hard coded fail-safe logic to take over
Overfitting of the YOLO model (poor performance in real world scenarios)	Detection failures in real world situations	Apply dataset augmentation and validation tests.	The IR+RF hybrid backup system is used to rely on.
Mechanical placement error in the model on Table Top.	Lack of lane detection and instability of the system	Correct CAD layout planning, prior to assembly.	Adjust the sensor/camera position and re-calibrate system again

7.5 Conclusion :

From the development of the Smart Traffic Management System, it can be seen that it has been well designed to be an intelligent and reliable system for real-time traffic control at a real 4-way intersection. It employs a hybrid system that integrates a computer vision module (YOLO) with Raspberry Pi and an IoT system (IR sensors and RF communication via Arduino Uno) for continuous operation in case of partial failure. The project has been realized with a total cost of 19,387 BDT fulfilling the practical requirements of a prototype of an embedded system with AI aspect. The system is designed for reliable detection of vehicles per lane, identification of priority vehicles (such as police and ambulance), and detection of wrong-way vehicles through systematic integration and testing of the critical components: USB camera, Raspberry Pi, custom-trained YOLO model, IR sensor network, RF modules, Arduino Uno, and traffic LEDs.

The table-top prototype and structured decision logic guarantees that based on real-time priority rules, only one lane gets a green signal at any given time. Systems with subsystem redundancy, the ability to switch between AI and sensor-based controls, and independent safety modules offer significant improvements in reliability when system components, such as the camera, processing time, or sensors, fail. Overall, the system offers an intelligent traffic control framework with low cost, fault-tolerant and high adaptability to future smart city applications, which is able to improve traffic efficiency, ensure emergency prioritization, and enhance road safety.

Chapter 8: Economical Analysis [CO12]

8.1 Introduction

In the case of a project, the economic assessment is one of the important factors to consider for the feasibility, sustainability and long term benefits. Economic analysis is the systematic study of the costs, investments, efficiency of the system and the expected returns. It assists in comprehending the allocation of resources, and the performance of the system in solving real-world problems.

For the Smart Traffic Management System, economic analysis involves considering the cost of initial installation, operational costs, and long-term savings in terms of fuel and reduced traffic congestion, as well as improving the efficiency of emergency services. Combination of Artificial Intelligence and IoT technologies can provide a cost effective and scalable solution especially for the developing regions.

This analysis enables the system to be compared to its cost and to determine if it is of sufficient value to be deployed on a large scale. This cost-benefit relationship emphasizes the economic benefits of using intelligent traffic systems as compared to conventional fixed-timing traffic control systems.

8.2 Economic Analysis

Proposal for Smart Traffic Management System is based on hybrid approach that combines AI based vision system and IoT based sensor system that decrease the reliance on costly infrastructure while keeping the system efficient and reliable. This system is low cost and decentralized, while traditional intelligent traffic systems have relied on high-end hardware and centralized control units.

The cost of the system at time of installation:

Table 16: The prototype development cost includes the following hardware components

Component	**Quantity**	**Unit Price (BDT)**	**Total (BDT)**
Arduino Uno R3	2	900	1800
USB Camera	4	520	2080
Raspberry Pi 5 (4GB RAM)	1	13000	13000
IR Sensors	8	45	360
RF Transmitter & Receiver	1 set	257	257
Relay Module (4-channel)	1	350	350
Traffic LEDs	4 sets	150	600
Buzzer	1	50	50
Power Supply	1	390	390
Miscellaneous Components			500

Total Hardware Cost = ৳19,387

No new computers/laptop was purchased for the prototype, as the system leverages the existing computing resources for the training and processing of the AI model.

Operational Cost:

The cost of the system for operation is relatively small because of:

- Consuming less power for Raspberry Pi and Arduino devices
- Minimal maintenance requirements
- No software tool cost as Python, OpenCV, YOLO (You Only Look Once) are used.
- The system has been engineered to run continuously with the minimum amount of human intervention, thereby saving a considerable amount of labour.

Economic Benefits:

This system offers a number of economic benefits:

1. Less Traffic Congestion. Dynamic signal control decreases delay at intersections for more efficient traffic flow.

2. Fuel Cost Savings: Less idling at traffic signals means lower fuel use and fuel cost savings for drivers.
3. Improved Emergency Response: Priority based signal control provides a faster emergency vehicle movement, which could save lives and costs.
4. Environmental Benefits: Clearing traffic congestion helps cut down emissions which helps ensure a greener environment.

Cost-Benefit Consideration:
The first prototype price is ৳19,387 but the system provides:

- Long-term operational savings
- Improved traffic efficiency
- Minimized reliance on manual traffic control
- The benefits, when scaled to actual implementation, far exceed the initial investment making the system economically viable.

8.3 Conclusion

The Smart Traffic Management System is economically feasible with its low implementation cost and high operational efficiency. The adoption of low-cost hardware and open-source software cuts costs and provides a powerful asset in traffic control and safety.
The system is sustainable and scalable, and can be implemented in urban and semi-urban areas, thereby contributing to the long-term economic and social development.

Chapter 9: Ethics and Professional Responsibilities [CO13, CO2]

9.1 Introduction:

Ethics and professional responsibility are key components of engineering projects, especially in the areas of automation, artificial intelligence, and public safety systems. Ethical aspects are crucial in the Smart Traffic Management System as it directly affects road users, emergency services, and traffic safety. The system automatically manages traffic signals and decision-making processes, which demands that the engineers design the system so that it is accurate, fair, reliable and transparent. This chapter covers the ethical issues related to the design, development, and deployment of the system.

9.2 Identify Ethical Issues and Professional Responsibility

Several ethical issues related to the use of AI and automated decision-making in the system:

1. **Accuracy and Reliability:** If the vehicle is not detected correctly or the AI system misclassifies the vehicle type, it can result in the incorrect management of the lights, potentially leading to congestion or safety hazards.
2. **System Failure Risks:** An inability to identify emergency vehicles could mean a delay in necessary services, which can have serious consequences.
3. **Privacy Concerns:** The use of camera-based detection systems can potentially be seen as a form of surveillance and misuse of information as they capture public data.
4. **Bias in AI Models:** The AI model can have varying performances in different conditions such as low light and weather, causing varying results.

Professional responsibilities include:

- Planning for system safety and reliability
- Giving precise and tested solutions
- Explicitly stating some of the system's limitations
- Implementing fail-safe mechanisms

9.3 Apply Ethical Issues and Professional Responsibility

Throughout the system design ethical considerations were applied:

- A hybrid system (AI + IoT backup) guarantees reliability even in the event of one system's failure.
- Wide range of tests are done to enhance detection accuracy.
- Arduino based backups are used to achieve fail-safety.
- Design of systems to minimise storage of unnecessary data to reduce privacy risks
- The system is focused on public safety and reliability, with ethical compliance in the real world.

9.4 Ethical Use of Automation and AI

Automation can make things more efficient, but there is a time and place for it. Vehicle detection by using YOLO (You Only Look Once) is allowed and will be used for real-time vehicle detection, however it will be:

- Continuously monitored
- With backup systems in place
- Human oversight is needed to make sure that automated decisions do not result in unsafe conditions.

9.5 Data Privacy and Transparency

The system requires visual information from traffic situations, thus creating privacy issues.
To address this:

- No long-term data storage is required – It's real-time processing data.
- There is no collection of personal identification data.
- System operations are transparent and explainable
- The system is transparent to the users and authorities, building trust and accountability.

9.6 Professional Accountability

Engineers involved in this project are responsible for:

- Maintaining the operation and security of systems
- Maintaining accurate documentation
- Taking timely action to fix system failures
- System performance is continually enhanced
- Professional accountability provides a reliable and trustworthy system over time.

9.7 Conclusion

The implementation of the Smart Traffic Management System is crucial and requires ethical design and professional responsibility. The system is accurate, transparent, safe, and accountable, thereby promoting positive values of engineering, which lead to positive contributions in society.

Chapter 10: Conclusion and Future Work

10.1 Project Summary / Conclusion

Smart Traffic Management System is a hybrid intelligent solution for a four-way intersection traffic control. The combination of AI-powered vehicle identification and IoT-enabled backup mechanisms effectively tackles critical traffic management issues of today.
The system demonstrates:

- Analysis of traffic density in real time.
- Emergency vehicle prioritization
- Detection and safety for wrong way
- Backup Systems for fail safe operation.

These features work together to form an effective, efficient, and scalable traffic management solution. The project underscores the value of using modern technologies in the improvement of urban infrastructure and in road safety.

10.2 Future Work

There are a number of enhancements and extensions that can be made to the system:

1. Advanced AI Integration: Improve performance in different conditions using more efficient and lightweight models.
2. Smart City Integration: Integrate the system with the centralized traffic management systems for enhanced coordination.
3. GPS-Based Emergency Detection: Switch RF modules for GPS tracking – more accurate, scalable.
4. Mobile Application Development: Give real time traffic information and system monitoring via mobile platforms.
5. Renewable Energy Integration: Install solar energy systems to make the solution more sustainable.
6. Data Analytics and Prediction: Predict traffic congestion based on traffic flow and adjust signals accordingly.

The future evolution of this system will enable it to become a complete intelligent traffic management solution that helps make a major contribution to smart city and sustainable urban development.

Chapter 11: Identification of Complex Engineering Problems and Activities

11.1 Identification of Attributes of Complex Engineering Problem (EP):

The BAETE framework defines a complex engineering problem as having advanced knowledge, multiple constraints, analysis, uncertainty and interaction between subsystems.

The proposed smart traffic management system possesses several important attributes of complex engineering problems:

- **P1: Depth of knowledge**
  Artificial Intelligence (YOLOv8), Embedded Systems, and Control Logic knowledge is required for the project. It is multidisciplinary and comprises integration of software and hardware.
- **P2: Range of Conflicting Requirements** - All requirements have to be completed.
  There must be a balance between the cost of the system (around ৳19387), the accuracy of detection, speed of processing and reliability. An engineering trade-off occurs when an improvement in one aspect impacts another.
- **P3: Level of Analysis Required**
  This project includes training a machine learning model, traffic density analysis, and designing FSM-based signal control. It also contains a comparison of the various design approaches.
- **P4: Knowledge of the issues**
  It is not a common problem. The traffic situation is very flexible and unpredictable and the solution needed individual design and testing.
- **P5: Codes to be applied: Limited**
  The standards like IEEE, IEC, ISO, BRTA guidelines are considered in the system, but full compliance with the standards is not implemented because it is prototype system.
- **P6: Stakeholder Involvement**
  The system has various actors including drivers, emergency services, authorities etc. and their needs might be conflicting.
- **P7: Interdependence**
  The system features modules that are interconnected such as the vision system, IoT backup, and signal controller. These components are interdependent in order to function.

The project in total possesses attributes P1 and some others, and therefore can be considered as a complex engineering problem.

11.2 Reasoning: How the Project Addresses EP Attributes

Integration, analysis and real-world constraints show the complexity of the project.

- It is a multidisciplinary system that involves integration of AI, embedded systems and control logic.

- It is pragmatic in terms of cost, performance and reliability trade-offs.
- Analytical techniques are used, like model training, congestion estimation and FSM control.
- The problem is open-ended, so it did not involve a direct application of theory but rather custom design.
- Standards are taken into account in design but not fully realised.
- Multiple stakeholders and their needs are addressed in the system design.
- The components in the system are dependent on each other, and they need to be coordinated and handled for any fault

11.3 Identification of Attributes of Complex Engineering Activities (EA)

The design, implementation and evaluation with up to date tools and for real-world constraints are involved in complex engineering activities.

- **A1: Broad range of resources:**
  The project uses various tools such as YOLOv8, OpenCV, Arduino, sensors, and microcontrollers.
- **A2: Level of Interaction:**
  It involves working together between software (AI), hardware (sensors and controllers), and system logic.
- **A3: Innovation and Creativity:**
  The hybrid architecture combining vision and IoT-based backup improves system reliability.
- **A4: Consequences and Impact:**
  Traffic efficiency, safety and emergency response are important factors of the society that are affected by the system.
- **A5: Familiarity of Activity:**
  The project includes non-routine engineering tasks, design, testing, and optimization.

11.4 Reasoning: How the Project Addresses EA Attributes

The project illustrates complex engineering activities within the context of modern engineering tools, system integration and approaches to real-world problem solving:

- The project is based on modern engineering tools and technologies.
- Needs to integrate and coordinate across subsystems.
- A hybrid solution is devised to enhance system performance and reliability.
- The solution is relevant from a traffic and safety point of view.
- The development process includes research, experimentation and performance evaluation.

References:

[1] S. Kamijo, Y. Matsushita, K. Ikeuchi, and M. Sakauchi, “Traffic monitoring and accident detection at intersections,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 2, pp. 108–118, Jun. 2000.

[2] R. Cucchiara, M. Piccardi, and P. Mello, “Image analysis and rule-based reasoning for a traffic monitoring system,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 2, pp. 119–130, Jun. 2000.

[3] K. Wang, Z. Li, Q. Yao, W. Huang, and F. Wang, “An intelligent traffic signal control decision support system for urban road intersections,” *Expert Systems with Applications*, vol. 37, no. 3, pp. 2148–2153, Mar. 2010.

[4] M. A. Hannan, A. Hussain, S. A. Samad, and P. J. Ker, “A review on vision-based vehicle detection, tracking, and behavior analysis in intelligent transportation systems,” *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 2, pp. 1–14, 2018.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 779–788.

[6] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “YOLOv4: Optimal speed and accuracy of object detection,” *arXiv preprint arXiv:2004.10934*, 2020.

[7] G. Bradski, “The OpenCV library,” *Dr. Dobb’s Journal of Software Tools*, vol. 25, no. 11, pp. 120–126, 2000.

[8] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Kauai, HI, USA, 2001, pp. I-511–I-518.

[9] M. M. Rashid, M. A. Rahman, and M. S. Hossain, “IoT based smart traffic management system for emergency vehicle clearance,” in *Proceedings of the International Conference on Robotics, Electrical Signal Processing and Techniques (ICREST)*, Dhaka, Bangladesh, 2019, pp. 489–494.

[10] S. Sharma, S. Pithora, G. Gupta, M. Goel, and M. Sinha, “Traffic light priority control for emergency vehicle using RFID,” *International Journal of Innovative Engineering and Technology*, vol. 2, no. 2, pp. 363–366, 2013.

[11] P. S. R. Diniz, E. A. B. Da Silva, and S. L. Netto, *Digital Signal Processing: System Analysis and Design for Smart Transportation Applications*. Cambridge, U.K.: Cambridge University Press, 2010.

[12] L. A. Klein, *Sensor Technologies and Data Requirements for ITS*. Boston, MA, USA: Artech House, 2001.

[13] H. Dia, "An agent-based approach to modelling driver route choice behaviour under the influence of real-time information," *Transportation Research Part C*, vol. 10, nos. 5–6, pp. 331–349, 2002.

[14] M. Papageorgiou, E. Kosmatopoulos, and I. Papamichail, "Effects of variable speed limits on motorway traffic flow," *Transportation Research Record*, vol. 2047, pp. 37–48, 2008.

[15] N. H. Gartner, C. J. Messer, and A. K. Rathi, *Traffic Flow Theory: A State-of-the-Art Report*. Washington, DC, USA: Transportation Research Board, 2001.

[16] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4–37, Jan. 2000.

[17] K. Ogata, *Modern Control Engineering*, 5th ed. Upper Saddle River, NJ, USA: Prentice Hall, 2010.

[18] R. C. Dorf and R. H. Bishop, *Modern Control Systems*, 13th ed. Boston, MA, USA: Pearson, 2016.

[19] Raspberry Pi Foundation, "Raspberry Pi hardware documentation," Cambridge, U.K., 2023.

[20] Arduino, "Arduino Uno Rev3 technical specifications," *Arduino Documentation*, 2023.

[21] T. S. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2002.

[22] A. Goldsmith, *Wireless Communications*. Cambridge, U.K.: Cambridge University Press, 2005.

[23] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Hoboken, NJ, USA: Pearson, 2021.

[24] M. Wooldridge, *An Introduction to MultiAgent Systems*, 2nd ed. Hoboken, NJ, USA: Wiley, 2009.

[25] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 4th ed. Belmont, MA, USA: Athena Scientific, 2017.

[26] Project Management Institute, *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*, 7th ed. Newtown Square, PA, USA: PMI, 2021.

[27] IEEE, *IEEE Code of Ethics*. Piscataway, NJ, USA: IEEE, 2023.

[28] National Society of Professional Engineers, *NSPE Code of Ethics for Engineers*. Alexandria, VA, USA: NSPE, 2019.

[29] *ISO 26262, Road Vehicles – Functional Safety*. Geneva, Switzerland: International Organization for Standardization, 2018.

[30] World Health Organization, *Global Status Report on Road Safety*. Geneva, Switzerland: WHO, 2023.

[31] United Nations, *Transforming Our World: The 2030 Agenda for Sustainable Development*. New York, NY, USA: UN, 2015.

[32] M. Keating, T. Quigley, J. Pohl, and Y. Zhang, *Sustainable Infrastructure: Principles into Practice*. Hoboken, NJ, USA: Wiley, 2013.

[33] J. Heizer, B. Render, and C. Munson, *Operations Management*, 13th ed. Boston, MA, USA: Pearson, 2020.

[34] S. M. Ross, *Introduction to Probability Models*, 12th ed. Amsterdam, Netherlands: Elsevier, 2019.

[35] R. Bajwa, R. Rajagopal, P. Varaiya, and R. Kavaler, "In-pavement wireless sensor network for vehicle classification," in *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*, Nashville, TN, USA, 2010, pp. 85–96.

[36] M. S. Islam, N. Z. Jhanjhi, and M. Humayun, "Smart city and intelligent transportation systems: A review using IoT and AI," *Sustainable Cities and Society*, vol. 72, 2021.

[37] S. C. Ergen and P. Varaiya, "PEDAMACS: Power efficient and delay aware medium access protocol for sensor networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 7, pp. 920–930, Jul. 2006.

[38] A. Hegyi, B. De Schutter, and H. Hellendoorn, "Optimal coordination of variable speed limits to suppress shock waves," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 1, pp. 102–112, Mar. 2005.

Appendix:

Code:

System code:

```
from google.colab import drive
drive.mount('/content/drive')
!pip install ultralytics opencv-python-headless

import os
import time
import cv2
from ultralytics import YOLO

try:
    from IPython.display import display, clear_output
    from PIL import Image
    IN_NOTEBOOK = True
except:
    IN_NOTEBOOK = False

# ---------- CONFIG ----------
BEST_PT = '/content/drive/MyDrive/System/Final-best.pt'

VIDEO_PATHS = {
    'A': '/content/drive/MyDrive/System/Normal_Video_720_new/Lane-A.mp4',
    'B': '/content/drive/MyDrive/System/Normal_Video_720_new/Lane-B.mp4',
    'C': '/content/drive/MyDrive/System/Normal_Video_720_new/Lane-C.mp4',
    'D': '/content/drive/MyDrive/System/Normal_Video_720_new/Lane-D.mp4',
}

OUTPUT_VIDEO =
'/content/drive/MyDrive/System/Outputs/normal-_vehicles-traffic-outputs.mp4'
TARGET_MOSAIC_WIDTH = 1280

CONFIDENCE = 0.25
IMGSZ = 640

# FSM timing (seconds)
GREEN_DURATION = 5
YELLOW_DURATION = 3
```

```
# Emergency robustness
EMERGENCY_MEMORY_TIME = 1.0
EMERGENCY_HOLD_TIME = 6.0

CLASS_NAMES = ["EmergencyVehicle", "Vehicle"]
EMERGENCY_CLASSES = ["EmergencyVehicle"]

# ---------- VIDEO UTIL ----------
class LoopingVideo:
    def __init__(self, path):
        self.cap = cv2.VideoCapture(path)
        if not self.cap.isOpened():
            raise FileNotFoundError(path)
        self.fps = self.cap.get(cv2.CAP_PROP_FPS) or 15.0

    def read(self):
        ok, frame = self.cap.read()
        if not ok:
            self.cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
            ok, frame = self.cap.read()
        return ok, frame

    def release(self):
        self.cap.release()

def draw_status(frame, lane, state, count, emergency=False):
    h, w = frame.shape[:2]
    cv2.rectangle(frame, (0, 0), (w, 60), (0, 0, 0), -1)

    color = (0,255,0) if state=='GREEN' else (0,255,255) if state=='YELLOW' else (0,0,255)
    cv2.circle(frame, (40,30), 15, (255,255,255), 2)
    cv2.circle(frame, (40,30), 12, color, -1)

    text = f"Lane {lane} | {state} | Vehicles: {count}"
    if emergency:
        text += " 🚨"

    cv2.putText(frame, text, (70,38),
                cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255,255,255), 2)
```

```
    return frame

def letterbox(img, shape):
    h,w = img.shape[:2]
    r = min(shape[0]/h, shape[1]/w)
    nh,nw = int(h*r), int(w*r)
    img = cv2.resize(img,(nw,nh))
    top = (shape[0]-nh)//2
    left = (shape[1]-nw)//2
    return cv2.copyMakeBorder(img, top, shape[0]-nh-top,
                              left, shape[1]-nw-left,
                              cv2.BORDER_CONSTANT,(0,0,0))

def make_mosaic(frames, width):
    order = ['A','B','C','D']
    panel_w = width//2
    panel_h = int(panel_w*9/16)
    imgs = [cv2.resize(letterbox(frames[k],(panel_h,panel_w)),
                       (panel_w,panel_h)) for k in order]
    return cv2.vconcat([cv2.hconcat(imgs[:2]), cv2.hconcat(imgs[2:])])

# ---------- FSM CONTROLLER ----------
class TrafficController:
    def __init__(self, lanes):
        self.lanes = lanes
        self.current_lane = lanes[0]
        self.state = 'GREEN'
        self.state_start = 0

        self.mode = 'NORMAL'
        self.emergency_lane = None
        self.emergency_start = 0

    def _next_lane(self, counts):
        if counts:
            return max(counts, key=counts.get)
        idx = (self.lanes.index(self.current_lane)+1)%len(self.lanes)
        return self.lanes[idx]

    def update(self, counts, emergencies, now):
```

```
# -------- EMERGENCY ENTRY --------
if emergencies and self.mode != 'EMERGENCY':
    lane = list(emergencies.keys())[0]
    self.mode = 'EMERGENCY'
    self.emergency_lane = lane
    self.emergency_start = now
    self.current_lane = lane
    self.state = 'GREEN'
    self.state_start = now
    return self.current_lane, 'GREEN', True

# -------- EMERGENCY HOLD --------
if self.mode == 'EMERGENCY':
    if emergencies or now - self.emergency_start < EMERGENCY_HOLD_TIME:
        return self.current_lane, 'GREEN', True

    # Emergency ended → force YELLOW
    self.mode = 'NORMAL'
    self.state = 'YELLOW'
    self.state_start = now
    return self.current_lane, 'YELLOW', False

# -------- NORMAL FSM --------
elapsed = now - self.state_start

if self.state == 'GREEN' and elapsed >= GREEN_DURATION:
    self.state = 'YELLOW'
    self.state_start = now

elif self.state == 'YELLOW' and elapsed >= YELLOW_DURATION:
    self.state = 'RED'
    self.state_start = now

elif self.state == 'RED':
    self.current_lane = self._next_lane(counts)
    self.state = 'GREEN'
    self.state_start = now

return self.current_lane, self.state, False
```

```
# ---------- MAIN ----------
assert os.path.exists(BEST_PT)
model = YOLO(BEST_PT)

readers = {k:LoopingVideo(v) for k,v in VIDEO_PATHS.items()}
fps = min([r.fps for r in readers.values()]+[15])

panel_w = TARGET_MOSAIC_WIDTH//2
panel_h = int(panel_w*9/16)

writer = cv2.VideoWriter(
    OUTPUT_VIDEO,
    cv2.VideoWriter_fourcc(*'mp4v'),
    fps,
    (TARGET_MOSAIC_WIDTH,panel_h*2)
)

controller = TrafficController(list(VIDEO_PATHS.keys()))
emergency_memory = {}
start_time = time.time()
frame_idx = 0

print("Simulation started...")

try:
    while True:
        now = time.time()-start_time
        frames, counts, emergencies = {}, {}, {}

        for lid, reader in readers.items():
            ok, frame = reader.read()
            if not ok: continue

            res = model.predict(frame, conf=CONFIDENCE, imgsz=IMGSZ, verbose=False)[0]
            count = 0
            detected = False

            if res.boxes:
                for b in res.boxes:
```

```
                cls = CLASS_NAMES[int(b.cls[0])]
                count += 1
                x1,y1,x2,y2 = b.xyxy[0].cpu().numpy().astype(int)
                col = (0,0,255) if cls in EMERGENCY_CLASSES else (0,255,0)
                cv2.rectangle(frame,(x1,y1),(x2,y2),col,2)
                cv2.putText(frame,cls,(x1,y1-5),
                            cv2.FONT_HERSHEY_SIMPLEX,0.6,(255,255,255),2)
                if cls in EMERGENCY_CLASSES:
                    detected = True

            counts[lid] = count
            if detected:
                emergency_memory[lid] = now
                        if lid in emergency_memory and now-emergency_memory[lid] <=
EMERGENCY_MEMORY_TIME:
                emergencies[lid] = True

            frames[lid] = frame

        lane, state, _ = controller.update(counts, emergencies, now)

        vis = {}
        for lid in frames:
            s = state if lid==lane else 'RED'
            vis[lid] = draw_status(frames[lid], lid, s, counts[lid], lid in emergencies)

        mosaic = make_mosaic(vis, TARGET_MOSAIC_WIDTH)
        writer.write(mosaic)

        if IN_NOTEBOOK and frame_idx%3==0:
            clear_output(wait=True)
            display(Image.fromarray(cv2.cvtColor(mosaic,cv2.COLOR_BGR2RGB)))

        frame_idx += 1

except KeyboardInterrupt:
    print("Stopped")

finally:
    for r in readers.values():
```

```
        r.release()
    writer.release()
    cv2.destroyAllWindows()
    print(f"✅ Saved: {OUTPUT_VIDEO}")
```

Train Code:

```
DATA_YAML = "/content/drive/MyDrive/FYDP_C/our-new-dataset-test-3-v3/jb-t3-v3-roboflow-data.yaml"
PROJECT_DIR = "/content/drive/MyDrive/yolo_runs"
MODEL_NAME = "traffic_yolov8m_1024"
# ==================

from ultralytics import YOLO

model = YOLO("yolov8m.pt")

model.train(
    data=DATA_YAML,
    epochs=80,
    imgsz=1024,
    batch=4,
    project=PROJECT_DIR,
    name=MODEL_NAME,
    save=True,
    resume=False
)
```

Arduino code:
Main Arduino:

```
// ---------------------- LED PINS -----------------------
const int A_RED    = 2;
const int A_GREEN  = 3;

const int B_RED    = 4;
const int B_GREEN  = 5;

const int C_RED    = 6;
```

```
const int C_GREEN  = 7;

const int D_RED    = 8;
const int D_GREEN  = 9;

// ---------------- COMMON YELLOW PIN --------------------
const int YELLOW_PIN = 10;  // all roads share this

// --------------------- IR SENSOR PINS ------------------
const int A1_IR = 11;
const int A2_IR = 12;

const int B1_IR = 13;
const int B2_IR = A0;

const int C1_IR = A1;
const int C2_IR = A2;

const int D1_IR = A3;
const int D2_IR = A4;

// ---------------- EMERGENCY BUTTON --------------------
const int EMERGENCY_PIN = A5;

// ------------------- TIMINGS --------------------------
const unsigned long GREEN_TIME_MS  = 4000;
const unsigned long YELLOW_TIME_MS = 2000;

// ------------------- ANALOG THRESHOLD -----------------
const int ANALOG_THRESHOLD = 500;


// -------------------------------------------------------
// SETUP
// -------------------------------------------------------
void setup() {

  Serial.begin(9600);

  // LEDs
```

```
  pinMode(A_RED, OUTPUT);   pinMode(A_GREEN, OUTPUT);
  pinMode(B_RED, OUTPUT);   pinMode(B_GREEN, OUTPUT);
  pinMode(C_RED, OUTPUT);   pinMode(C_GREEN, OUTPUT);
  pinMode(D_RED, OUTPUT);   pinMode(D_GREEN, OUTPUT);
  pinMode(YELLOW_PIN, OUTPUT);

  // IR pins
  pinMode(A1_IR, INPUT);
  pinMode(A2_IR, INPUT);
  pinMode(B1_IR, INPUT);
  pinMode(B2_IR, INPUT);

  pinMode(EMERGENCY_PIN, INPUT);

  setAllRed();

  Serial.println("SMART Traffic System Ready...");
}


// ------------------------------------------------------
// SET ALL SIGNALS TO RED
// ------------------------------------------------------
void setAllRed() {
  digitalWrite(A_RED, HIGH); digitalWrite(A_GREEN, LOW);
  digitalWrite(B_RED, HIGH); digitalWrite(B_GREEN, LOW);
  digitalWrite(C_RED, HIGH); digitalWrite(C_GREEN, LOW);
  digitalWrite(D_RED, HIGH); digitalWrite(D_GREEN, LOW);

  digitalWrite(YELLOW_PIN, LOW);
}


// ------------------------------------------------------
// READ DIGITAL SENSOR
// ------------------------------------------------------
int readDigitalSensor(int pin) {
  return digitalRead(pin) == HIGH ? 1 : 0;
}
```

```
// ------------------------------------------------
// READ ANALOG SENSOR → DIGITAL
// ------------------------------------------------
int readAnalogSensorAsDigital(int pin) {
  return (analogRead(pin) > ANALOG_THRESHOLD) ? 1 : 0;
}


// ------------------------------------------------
// GET ROAD DECISION (C1, C2, C3)
// ------------------------------------------------
int getDecision(int s1, int s2, bool analogMode=false) {

  int v1 = analogMode ? readAnalogSensorAsDigital(s1) : readDigitalSensor(s1);
  int v2 = analogMode ? readAnalogSensorAsDigital(s2) : readDigitalSensor(s2);

  if (v1 == 1 && v2 == 1) return 1;   // empty
  if (v1 == 0 && v2 == 0) return 3;   // heavy jam
  return 2;                           // medium jam
}


// ------------------------------------------------
// ALLOW ROAD (GREEN → YELLOW → RED)
// Green ON → Red forced OFF
// ------------------------------------------------
void allowRoad(char road) {

  setAllRed();  // safety

  // Emergency override first
  if (analogRead(EMERGENCY_PIN) > 800) {
    Serial.println("⚠️ EMERGENCY ACTIVE → C ROAD ONLY GREEN");

    setAllRed();
    digitalWrite(C_RED, LOW);
    digitalWrite(C_GREEN, HIGH);
    delay(200);
    return;
```

```
  }

  // ---------- GREEN ON (RED OFF) ----------
  if (road == 'A') {
    digitalWrite(A_RED, LOW);
    digitalWrite(A_GREEN, HIGH);
  }
  else if (road == 'B') {
    digitalWrite(B_RED, LOW);
    digitalWrite(B_GREEN, HIGH);
  }
  else if (road == 'C') {
    digitalWrite(C_RED, LOW);
    digitalWrite(C_GREEN, HIGH);
  }
  else if (road == 'D') {
    digitalWrite(D_RED, LOW);
    digitalWrite(D_GREEN, HIGH);
  }

  delay(GREEN_TIME_MS);

  // GREEN OFF before yellow
  if (road == 'A') digitalWrite(A_GREEN, LOW);
  if (road == 'B') digitalWrite(B_GREEN, LOW);
  if (road == 'C') digitalWrite(C_GREEN, LOW);
  if (road == 'D') digitalWrite(D_GREEN, LOW);

  // ---------- YELLOW ----------
  digitalWrite(YELLOW_PIN, HIGH);
  delay(YELLOW_TIME_MS);
  digitalWrite(YELLOW_PIN, LOW);

  // back to RED
  setAllRed();
}


// -------------------------------------------------------
// MAIN LOOP
```

```
// ------------------------------------------------------
void loop() {

  // Emergency first priority
  if (analogRead(EMERGENCY_PIN) > 800) {
    Serial.println("⚠️ EMERGENCY: Forcing C Road GREEN");
    setAllRed();
    digitalWrite(C_RED, LOW);
    digitalWrite(C_GREEN, HIGH);
    delay(200);
    return;
  }

  // Read decisions
  int A_dec = getDecision(A1_IR, A2_IR);
  int B_dec = getDecision(B1_IR, B2_IR);
  int C_dec = getDecision(C1_IR, C2_IR, true);
  int D_dec = getDecision(D1_IR, D2_IR, true);

  // Priority C3
  if (A_dec == 3) { allowRoad('A'); return; }
  if (B_dec == 3) { allowRoad('B'); return; }
  if (C_dec == 3) { allowRoad('C'); return; }
  if (D_dec == 3) { allowRoad('D'); return; }

  // Priority C2
  if (A_dec == 2) { allowRoad('A'); return; }
  if (B_dec == 2) { allowRoad('B'); return; }
  if (C_dec == 2) { allowRoad('C'); return; }
  if (D_dec == 2) { allowRoad('D'); return; }

  // All empty → Normal cycle
  allowRoad('A');
  allowRoad('B');
  allowRoad('C');
  allowRoad('D');
}
```

Secondary Arduino:

```
// ================================
// Secondary Arduino
// Wrong-Way Vehicle Detection
// Using 2 IR Sensors + Buzzer
// ================================

// Pin Definitions
#define IR1 2      // First IR Sensor
#define IR2 3      // Second IR Sensor
#define BUZZER 8   // Buzzer

// Variables
bool ir1_triggered = false;
bool ir2_triggered = false;
bool wrong_way = false;

void setup() {
  pinMode(IR1, INPUT);
  pinMode(IR2, INPUT);
  pinMode(BUZZER, OUTPUT);

  digitalWrite(BUZZER, LOW); // Buzzer OFF initially
}

void loop() {

  // Read IR sensors (LOW means object detected)
  if (digitalRead(IR1) == LOW) {
    ir1_triggered = true;
    delay(100); // debounce
  }

  if (digitalRead(IR2) == LOW) {
    ir2_triggered = true;
    delay(100); // debounce
  }

  // -------- LOGIC CHECK --------
```

```
  // Correct direction: IR1 → IR2
  if (ir1_triggered && ir2_triggered && !wrong_way) {
    // Correct movement
    resetFlags();
    digitalWrite(BUZZER, LOW);
  }

  // Wrong direction: IR2 → IR1
  if (ir2_triggered && ir1_triggered && !wrong_way) {
    wrong_way = true;
    digitalWrite(BUZZER, HIGH); // Turn ON buzzer
  }

  // If wrong way active, wait until the correct order happens
  if (wrong_way) {
    if (digitalRead(IR1) == LOW) {
      delay(100);
      if (digitalRead(IR2) == LOW) {
        // Correct direction detected → stop buzzer
        wrong_way = false;
        digitalWrite(BUZZER, LOW);
        resetFlags();
      }
    }
  }
}

// Reset IR flags
void resetFlags() {
  ir1_triggered = false;
  ir2_triggered = false;
}
```

Logbook :

FYDP (P) Summary of Team Log Book:

Table 17: Log Book (FYDP-P)

Date/Time	Attendence	Summary of the Meetings	Responsible	Comment by ATC
11.02.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Initial brainstorming for project title. 2. Discuss initial problem statement ideas. 3. Plan draft concept note	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul, Task 3: Yusuf	Task 1 completed, Task 2 partially completed, Task 3 in progress
11.02.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Narrow down problem statement options. 2. Begin drafting concept note	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 in progress
14.02.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Finalize project title and problem statement. 2. Complete draft concept note	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 partially completed, Task 2 in progress
17.02.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Gather sources for literature review. 2. Assign initial reading tasks	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 in progress
18.02.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Conduct initial reading of gathered sources. 2. Start synthesizing notes	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 in progress
18.02.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Complete synthesis and notes from literature. 2. Prepare for market survey design	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 in progress
22.02.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Design roadside implementation questionnaire. 2. Plan data collection strategy	Task 1: Zayed, Task 2: Yusuf	Task 1 completed, Task 2 in progress

01.03.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Collect data regarding roadside implementation. 2. Begin initial analysis	Task 1: Zayed, Task 2: Yusuf	Task 1 completed, Task 2 partially completed
15.03.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Complete analysis of roadside implementation data. 2. Prepare draft report outline	Task 1: Zayed, Task 2: Yusuf	Task 1 completed, Task 2 in progress
25.03.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Create outline for draft note. 2. Assign content drafting tasks	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 in progress
01.04.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Draft content for project note. 2. Review initial drafts	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 partially completed
15.04.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Review and edit draft notes. 2. Prepare for design phase	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 partially completed
26.02.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Develop initial design concepts. 2. Start sketching prototypes	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 in progress
03.03.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Create initial sketches of prototypes. 2. Gather feedback on designs	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 in progress
08.03.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Refine prototype designs. 2. Plan risk assessment	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 in progress

12.03.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Identify potential project risks. 2. Evaluate identified risks	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 in progress
10.03.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Complete risk evaluation. 2. Plan mitigation strategies	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 in progress
13.03.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Develop mitigation plans for risks. 2. Discuss ethical considerations	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 partially completed
25.04.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Compile final proposal document	Zayed, Yusuf, Fahmidul and Ehsanul	Completed

FYDP (D) Summary of Team Log Book:

Table 18: Log Book (FYDP-D)

Date/Time	Attendence	Summary of the Meetings	Responsible	Comment by ATC
17.06.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Initial review of prototype designs. 2. Discuss adjustments based on feedback.	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 in progress
17.06.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Fixing appropriate software to build circuits. 2. Fixing suitable software for model train.	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 in progress
24.06.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Finalize refined prototype designs. 2. Prepare for simulation setup.	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 partially completed
03.07.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Set up simulation for Design 1 (Image processing). 2. Test initial parameters.	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 partially completed
08.07.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Run simulation tests for Design 1. 2. Document initial results.	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 partially completed, Task 2 in progress
15.07.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Analyze simulation results for Design 1. 2. Plan Design 2 simulation.	Task 1: Zayed, Task 2: Yusuf	Task 1 partially completed, Task 2 in progress
22.07.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Set up simulation for Design 2 (hybrid system). 2. Formulate IoT sensor functions with Design 1.	Task 1: Zayed, Task 2: Yusuf	Task 1 partially completed, Task 2 in progress

28.07.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Run tests for Design 2 simulation. 2. Troubleshoot.	Task 1: Zayed, Task 2: Yusuf	Task 1 partially completed, Task 2 in progress
01.08.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Analyze Design 2 simulation results. 2. Prepare data compilation.	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 in progress
10.08.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Compile data from both designs. 2. Begin initial analysis.	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 partially completed, Task 2 in progress
12.08.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Conduct initial analysis of results. 2. Plan detailed review.	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 in progress
20.08.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Prepare slides for the final presentation. 2. Draft initial report.	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 in progress
10.08.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Continue drafting the final report. 2. Review presentation content.	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 in progress
25.08.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Conduct final review of report and presentation. 2. Prepare for submission.	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 partially completed
28.08.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Submit final deliverables. 2. Review project outcomes.	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Completed

FYDP (C) Summary of Team Log Book:

Table 19: Log Book (FYDP-C)

Date/Time	Attendence	Summary of the Meetings	Responsible	Comment by ATC
07.10.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Initial meeting to discuss project scope. 2. Exchange ideas for real time roadside system	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 in progress
08.10.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Build consensus on project objectives. 2. Plan literature review.	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 in progress
12.10.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Review initial topic options. 2. Collect feedback from the team.	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 partially completed
15.10.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Finalize feedback collection. 2. Prepare for the final decision.	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 in progress
20.10.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Draft outline for concept note. 2. Assign content writing tasks.	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 partially completed, Task 2 in progress
25.10.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Rehearse progress presentation. 2. Plan for multiple design approaches	Task 1: Zayed, Task 2: Yusuf	Task 1 partially completed, Task 2 in progress
05.11.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Gather data for a real life system. 2. Analyze the final design approach	Task 1: Zayed, Task 2: Yusuf	Task 1 partially completed, Task 2 in progress

20.11.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Analyze gathered data. 2. Develop initial design.	Task 1: Zayed, Task 2: Yusuf	Task 1 partially completed, Task 2 in progress
25.11.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Set up criteria for optimal solution. 2. Begin evaluation process.	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 in progress
01.12.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Implement optimal design approach. 2. Plan budget estimation.	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 partially completed, Task 2 in progress
05.12.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Estimate costs for prototype. 2. Allocate resources.	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 in progress
10.12.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Finalize resource allocation. 2. Review ethical considerations.	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 in progress
10.12.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Plan safety measures. 2. Review initial draft.	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 in progress
12.12.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Review and gather feedback. 2. Prepare the final system	Task 1: Zayed and Yusuf, Task 2: Fahmidul and Ehsanul	Task 1 completed, Task 2 partially completed
15.12.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Conduct final troubleshooting. 2. Outline report structure	Task 1: Zayed, Task 2: Yusuf	Task 1 completed, Task 2 partially completed

20.12.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Perform final edits. 2. Review project outcomes	Task 1: Zayed, Task 2: Yusuf	Task 1 partially completed, Task 2 in progress
26.12.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Create outline for proposal report. 2. Begin content development	Task 1: Zayed, Task 2: Yusuf	Task 1 completed, Task 2 partially completed
28.12.2025	1. Yusuf 2. Zayed 3. Fahmidul 4. Ehsanul	1. Last moment rehearsal 2. Showcase presentation video making	Task 1: Zayed, Task 2: Yusuf	Completed the overall system engineering (ready for the showcase)