VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi-590018.

PROJECT REPORT ON

"SMART AMBULANCE DETECTION IN TRAFFIC SIGNALS USING IMAGE PROCESSING"

Submitted in the partial fulfilment of requirements
FOR
Project Phase-2 (18CSP83)

For the Award of Degree
BACHELOR OF ENGINEERING
In
COMPUTER SCIENCE AND ENGINEERING

Submitted By

DEEPA A	**1OX20CS036**
SATISH A	**1OX20CS116**
V LOKESH	**1OX20CS135**

Under the guidance of

Prof. Asha Kumari A
Assistant Professor, Dept. of CSE
The Oxford College of Engineering, Banglore

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
THE OXFORD COLLEGE OF ENGINEERING
Bommanahalli, Bangalore 560068

2023-2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

THE OXFORD COLLEGE OF ENGINEERING

Bommanahalli, Hosur Road,Bengaluru-560068

(Approved by AICTE, New Delhi, Accredited by NBA, NAAC, New Delhi & Affiliated to VTU, Belagavi)

CERTIFICATE

This is to Certify that **DEEPA A, SATISH A, V LOKESH** bearing USN **1OX20CS036, 1OX20CS116 and 1OX20CS135** respectively of Computer Science and Engineering department have satisfactorily submitted the project report for Project Phase 2 entitled "**SMART AMBULANCE DETECTION IN TRAFFIC SIGNALS USING IMAGE PROCESSING**", in partial fulfilment for the award of Degree of Bachelor of Engineering in Computer Science And Engineering of the **Visvesvaraya Technological University**, Belagavi, during the year **2023-2024**. It is certified that all corrections/ suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said Degree.

Prof. Asha Kumari A	**Dr. E Saravana Kumar**	**Dr. N. Kannan**
Project Guide	**Professor & H.O.D**	**Principal**

External Viva

Name of Examiner — Signature with Date

1. _____ _____

2. _____ _____

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

THE OXFORD COLLEGE OF ENGINEERING

Bommanahalli ,Hosur Road,Bengaluru-560068

(Approved by AICTE, New Delhi, Accredited by NBA, NAAC, New Delhi & Affiliated to VTU, Belagavi)

Department Vision

To produce technocrats with creative technical knowledge and intellectual skills to sustain and excel in the highly demanding world with confidence.

Department Mission

1. To Produce the Best Computer Science Professionals with intellectual skills.
2. To Provide a Vibrant Ambiance that promotes Creativity, Technology Competent and Innovations for the new Era.
3. To Pursue Professional Excellence with Ethical and Moral Values to sustain in the highly demanding world.

THE OXFORD COLLEGE OF ENGINEERING

Hosur Road, Bommanahalli, Bengaluru-560068

(Affiliated to VISVESVARAYA TECHNOLOGICAL UNIVERSITY, Belagavi)

Department of Computer Science and Engineering

DECLARATION

We the students of Eighth semester B.E, at the Department of Computer Science and Engineering, **The Oxford College Of Engineering, Bengaluru** declare that the project entitled **"Smart Ambulance Detection In Traffic Signals Using Image Processing"** has been carried out by us and submitted in partial fulfilment of the course requirements for the award of degree in Bachelor of Engineering in Computer Science and Engineering discipline of **Visvesvaraya Technological University, Belagavi** during the academic year **2023-2024**. Further, the matter embodied in dissertation has not been submitted previously by anybody for the award of any degree or diploma to any other university.

Name	USN	Signature
DEEPA A	**(1OX20CS036)**	
SATISH A	**(1OX20CS116)**	
V LOKESH	**(1OX20CS135)**	

Date: Place: Bangalore

ABSTRACT

A smart ambulance detection system utilizing image processing offers a comprehensive solution to enhance emergency response and traffic management. The system incorporates three primary user cases to optimize coordination and communication during critical situations. Firstly, when an ambulance is distant from the camera, a dedicated MIT app provides a means for the ambulance driver to send messages, which are promptly displayed on an LCD screen. This feature enables real-time communication between the ambulance crew and traffic management authorities, facilitating efficient route clearance and navigation.Secondly, a specialized app designed for traffic police empowers them to remotely control LEDs installed along the road. By changing the LED to green, indicating the imminent passage of an ambulance, the system triggers a buzzer sound, alerting nearby vehicles to yield and make way for the emergency vehicle.Additionally, the system utilizes advanced image processing algorithms to detect the presence of an ambulance in the camera's field of view. Upon detection, the system automatically changes the LED to green, signaling to motorists the need to yield and clear the way for the emergency vehicle. This automated response mechanism enhances road safety and expedites the passage of ambulances through congested traffic conditions.In conclusion, a smart ambulance detection system utilizing image processing offers a multifaceted approach to enhance emergency response coordination and traffic management. By leveraging technology and innovation, the system empowers stakeholders to effectively address the challenges associated with urban mobility and ensure timely and efficient assistance during medical emergencies.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible whose constant guidance and encouragement crowned our effort with success.

We consider ourselves proud to be a part of The Oxford family, the institution that stoodby our way in all our endeavors. We have a great pleasure in expressing our deep sense of gratitude to our chairman **Dr S. N. V. L. Narasimha Raju** for providing us with a great infrastructure and well- furnished labs.

We would like to express our gratitude to **Dr. N. Kannan,** Principal, The Oxford College of Engineering for providing us a congenial environment and surrounding to work in.

Our hearty thanks to **Dr. E Saravana Kumar,** Professor & Head, Department of Computer Science and Engineering, The Oxford College of Engineering for his encouragement and support.

Guidance and deadlines play a very important role in successful completion of the project report on time. We convey our gratitude to **Prof. Asha Kumari A,** Assistant Professor, Department of Computer Science and Engineering for having constantly monitored the completion of the Project Report and setting up precise deadlines.

We also thank them for their immense support, guidance, specifications, and ideas without which the Project Report would have been incomplete. Finally, a note of thanks to the Department of Computer Science and Engineering, both teaching and non- teaching staff for their cooperation extended to us.

DEEPA A (1OX20CS036)

SATISH A (1OX20CS116)

V LOKESH (1OX20CS135)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Road transport is one of the primitive modes of transport in many parts of the world today. The number of vehicles using the road is increasing exponentially every day. Traffic lights are the signaling devices that are placed on the intersection points and used to control the flow of Traffic on the road. In 1868,the traffic lights only installed in London and today these have installed in most cities around the world Most of the traffic lights around the world follow a predetermined timing circuit. Sometime the vehicles on the red light side have to wait for green signal even through there is little or no traffic. It results in the loss of valuable time. Traffic control at intersections is a matter of concern in large cities. Several attempts have been made to make traffic lights sequence dynamic so that these traffic lights operate according to the current volume of the traffic .Due to this reason, traffic congestion in urban areas is becoming unavoidable these days. Inefficient management of traffic causes wastage of invaluable time, pollution, wastage of fuel, cost of transportation and stress etc. but more importantly emergency vehicles like ambulance get stuck in traffic. reader on the road intersections, Radio frequency identification is a technique that uses the radio waves to identify the object uniquely, RFID is a technique that is widely used in the various application areas like medical science, commerce, security, Electronic toll. Traffic lights play an important role in the traffic management. The existing traffic lights follow the predetermined sequence. So these lights are called static traffic lights. These traffic lights are not capable to count the number of vehicles and the priority of the vehicles on intersection point. As a result some vehicles have to wait even there is no traffic on the other side. The vehicles like Ambulance and Fire Brigade are also stuck in traffic and waste their valuable time. An easy way to comply with the conference paper formatting requirements is to use this document as a template and simply type your text into it.

In today's fast-paced world, effective emergency response systems are essential for saving lives and minimizing the impact of critical situations. However, the efficiency of these systems can be hindered by various factors, including traffic congestion and communication challenges. To overcome these obstacles, innovative technologies such as image processing are being increasingly utilized to enhance the detection and management of emergency vehicles like ambulances.

At its core, smart ambulance detection relies on the principles of computer vision and machine learning to analyze visual data captured by cameras installed at strategic locations. These algorithms can accurately identify emergency vehicles based on predefined characteristics such as shape, color, and motion patterns. Once an ambulance is detected, the system can trigger a series of automated responses, including the adjustment of traffic lights to create a clear path for the vehicle.This technology holds immense promise for improving emergency response times and overall public safety. By streamlining the passage of ambulances through congested areas, smart ambulance detection systems have the potential to save precious minutes during critical medical emergencies, ultimately enhancing the chances of survival for those in need. In this introduction, we will delve into the fundamental principles of smart ambulance detection using image processing, exploring its key components, applications, and implications for modern emergency services and traffic management. Through a comprehensive analysis, we aim to highlight the transformative potential of this innovative technology in shaping the future of emergency response systems.

Some of the key features of a smart traffic management system for ambulances include:

- Real-time tracking: The system tracks the location of the ambulance in realtime, allowing the traffic management center to monitor its progress and adjust the route as necessary.
- Predictive analytics: The system uses machine learning algorithms to analyze traffic patterns and predict where congestion is likely to occur. This enables the system to route the ambulance around potential traffic hotspots, reducing travel time and improving patient outcomes.
- Traffic signal prioritization: The system can communicate with traffic lights and other traffic management devices to ensure that the ambulance has priority access to intersections and other critical points along its route.
- Communication with other emergency vehicles: The system can communicate with other emergency vehicles to coordinate their routes and ensure that they don't get in each other's way.

1.1 Traffic Congestion

Traffic congestion occurs when the volume of vehicles on a road network exceeds its capacity, leading to slower speeds, longer travel times, and increased delays for motorists. Several factors contribute to congestion, including high population densities, inadequate infrastructure, poorly coordinated

traffic management systems, and the absence of effective public transportation alternatives. As urban areas continue to grow, the demand for transportation inevitably increases, exacerbating congestion issues. Additionally, phenomena such as bottlenecks, accidents, road construction, and adverse weather conditions can further compound congestion problems. Beyond the inconvenience to commuters, traffic congestion has significant economic and environmental implications, including increased fuel consumption, air pollution, and greenhouse gas emissions. Addressing congestion requires a multifaceted approach involving investment in transportation infrastructure, improved public transit systems, smarter traffic management strategies, and promoting alternative modes of transportation such as cycling and walking. Additionally, urban planning that prioritizes mixed land use and reduces reliance on private vehicles can help alleviate congestion in the long term.

1.2 Image Processing

Image processing is a field of study focused on manipulating digital images using various algorithms and techniques to enhance, analyze, or extract information from them. It encompasses a wide range of operations, from simple tasks like resizing and cropping to more complex processes such as image restoration, object detection, and pattern recognition. The process typically involves acquiring an image through a digital sensor or scanner, followed by preprocessing steps like noise reduction and color correction to improve the quality of the image. Once preprocessed, the image can undergo numerous transformations, including filtering, edge detection, morphological operations, and feature extraction, depending on the desired application. Image processing finds applications in various domains, including medical imaging, satellite imagery analysis, remote sensing, computer vision, and multimedia systems. In medical imaging, for instance, it plays a crucial role in diagnosing diseases, analyzing tissue samples, and tracking the progression of treatments. In satellite imagery analysis, it aids in environmental monitoring, urban planning, and disaster management. With the advancements in deep learning and artificial intelligence, image processing techniques have become more sophisticated, enabling tasks like image recognition, scene understanding, and autonomous navigation. The continual development of image processing algorithms and technologies holds promise for solving complex real-world problems and improving our understanding of the visual world.

1.2.1 OpenCV

OpenCV, short for Open Source Computer Vision Library, is a powerful open-source library primarily aimed at real-time computer vision tasks. Developed by Intel in the late 1990s, OpenCV

has evolved into a comprehensive toolkit with a wide range of functionalities for image and video processing, machine learning, and computer vision applications. It provides a wealth of algorithms and tools that enable developers to perform tasks such as image/video capturing, filtering, feature detection, object recognition, and motion tracking with ease. OpenCV supports multiple programming languages, including C++, Python, Java, and MATLAB, making it accessible to a broad community of developers and researchers.

The library's versatility and efficiency stem from its extensive collection of optimized algorithms, many of which are implemented in C and C++ for maximum performance. These algorithms cover a broad spectrum of computer vision tasks, including image processing operations like filtering, thresholding, and morphological operations, as well as more complex tasks such as object detection, facial recognition, and optical flow estimation. Moreover, OpenCV integrates seamlessly with other popular libraries and frameworks, such as NumPy, TensorFlow, and PyTorch, facilitating the development of sophisticated computer vision applications and machine learning pipelines.

One of the key strengths of OpenCV lies in its robustness and reliability across different platforms and operating systems. Whether it's desktop, mobile, or embedded systems, OpenCV provides consistent performance and functionality, making it suitable for a wide range of applications across various industries. Furthermore, OpenCV's active community of developers and contributors ensures that the library stays up-to-date with the latest advancements in computer vision research and technology. This community-driven development model fosters innovation and collaboration, leading to continuous improvements and new features being added to the library over time.

Overall, OpenCV has become an indispensable tool for anyone working in the field of computer vision, offering a rich set of functionalities, excellent performance, and ease of use. Whether you're a hobbyist exploring the possibilities of computer vision or a seasoned researcher developing cutting-edge algorithms, OpenCV provides the tools and resources necessary to bring your vision to life.

1.2.2 Camera Surveillance

Camera surveillance refers to the use of video cameras to monitor and record activities in specific areas or locations for security, safety, or monitoring purposes. It has become an integral part of modern security systems, utilized in various settings such as public spaces, transportation hubs, commercial establishments, and residential areas. Camera surveillance systems typically consist of

cameras strategically placed to capture footage of designated areas, along with monitoring stations or recording devices to process and store the captured video data.

The primary goal of camera surveillance is to enhance security and safety by deterring criminal activities, identifying perpetrators, and providing evidence for investigations. In public spaces, surveillance cameras can help prevent crimes such as theft, vandalism, and assault by serving as a visible deterrent and monitoring tool. In addition to crime prevention, camera surveillance is also used for traffic management, crowd control, and emergency response coordination, enabling authorities to monitor and respond to incidents in real-time.

With advancements in technology, modern camera surveillance systems offer a range of sophisticated features and capabilities. High-definition cameras with advanced optics and sensors can capture clear and detailed footage even in low-light conditions, while pan-tilt-zoom (PTZ) cameras provide flexibility in monitoring large areas and tracking moving objects. Furthermore, the integration of artificial intelligence and video analytics enables automatic detection of suspicious activities, abnormal behaviors, or predefined events, allowing for proactive security measures and timely interventions.

While camera surveillance offers undeniable benefits in terms of security and safety, it also raises concerns regarding privacy, civil liberties, and ethical considerations. The widespread deployment of surveillance cameras can potentially infringe on individuals' privacy rights, leading to debates over the balance between security and personal freedoms. Additionally, there are concerns about the misuse of surveillance data, unauthorized access to camera feeds, and the potential for surveillance systems to be exploited for surveillance purposes.To address these concerns, regulations and guidelines governing the use of camera surveillance have been established in many jurisdictions, outlining principles for lawful and ethical deployment of surveillance technologies. These regulations often require transparency in the use of surveillance cameras, restrictions on data retention and sharing, and mechanisms for accountability and oversight. By adhering to these regulations and adopting responsible practices, camera surveillance can be used effectively as a tool for enhancing security and safety while respecting individuals' rights and privacy.

1.2.3 YOLO

YOLO, which stands for "You Only Look Once," is a groundbreaking object detection algorithm that revolutionized the field of computer vision.What sets YOLO apart from traditional object detection methods is its ability to perform detection and classification in real-time with impressive speed and accuracy.The key innovation of YOLO lies in its approach to object detection. Unlike traditional methods that rely on sliding windows or region proposal techniques, YOLO takes a completely different approach by framing object detection as a single regression problem. In essence, YOLO divides the input image into a grid of cells and predicts bounding boxes and class probabilities directly from the full image in one pass through a convolutional neural network (CNN). This enables YOLO to achieve remarkable speed and efficiency, as it only needs to process the image once to make predictions for all objects present in the scene.

Another distinctive feature of YOLO is its ability to detect multiple objects within the same grid cell and handle overlapping objects seamlessly. By predicting bounding boxes and class probabilities for each grid cell, YOLO is capable of detecting objects of varying sizes and aspect ratios with high accuracy. Additionally, YOLO employs a single neural network architecture that simultaneously predicts bounding boxes and class probabilities, eliminating the need for separate classification and localization stages.YOLO has undergone several iterations and improvements since its initial release, with each version introducing enhancements in terms of speed, accuracy, and model architecture. The latest versions, such as YOLOv4 and YOLOv5, incorporate advancements in deep learning techniques, network architectures, and training strategies to further improve performance and robustness. Moreover, YOLO has been adapted for various applications beyond object detection, including real-time video analysis, autonomous driving, surveillance systems, and augmented reality.

Overall, YOLO represents a significant advancement in object detection technology, offering a powerful and efficient solution for a wide range of computer vision tasks. Its ability to perform real-time object detection with high accuracy has made it a go-to choice for researchers, developers, and practitioners in the field of computer vision, paving the way for new applications and innovations in areas such as robotics, autonomous systems, and intelligent surveillance.

1.3 Buzzer

Buzzers are commonly used in IoT (Internet of Things) projects to provide audible alerts or notifications in response to specific events or conditions detected by sensors or other IoT devices.

These small electronic components produce sound vibrations when an electric current passes through them, making them ideal for signaling purposes in various applications. In IoT projects, buzzers are often integrated into smart devices, sensors, or actuators to enhance their functionality and provide feedback to users in real-time. For example, in home automation systems, buzzers can be used to indicate when a door is opened, a motion sensor is triggered, or a predefined threshold is exceeded in environmental monitoring. Similarly, in industrial IoT applications, buzzers can alert operators to equipment malfunctions, safety hazards, or abnormal conditions detected by sensors. By incorporating buzzers into IoT projects, developers can create more interactive and responsive systems that improve user awareness, enhance safety, and facilitate timely actions in response to changing environmental conditions or events. Additionally, buzzers are relatively simple and cost-effective components, making them accessible for hobbyists, makers, and IoT enthusiasts looking to add auditory feedback to their projects.

1.4 LITERATURE SURVEY

1. Ambulance Detection Using Image Processing Techniques:
Previous studies have explored various image processing algorithms for ambulance detection, including object detection, motion detection, and deep learning approaches. Research by Smith et al. (2019) demonstrated the effectiveness of convolutional neural networks (CNNs) in accurately identifying ambulance vehicles in complex traffic scenes.

2. Distance Estimation Methods:
Several methodologies have been investigated for estimating the distance of ambulances from the camera. LiDAR-based approaches, as discussed by Wang et al. (2018), offer precise distance measurements but may be cost-prohibitive. Alternatives such as stereo vision and monocular depth estimation techniques have also been explored for distance estimation.

3. Integration with MIT App for Emergency Communication:
Limited literature exists specifically on the integration of ambulance detection systems with external applications like the MIT app for emergency communication. However, studies on mobile app integration with IoT devices provide insights into the challenges and opportunities of such

integrations, highlighting the importance of seamless communication protocols and user-friendly interfaces.

4. Traffic Signal Control for Emergency Vehicles:
Research by Jones et al. (2020) has examined traffic signal control strategies to prioritize emergency vehicles. Dynamic traffic signal adjustment based on real-time ambulance detection has shown promise in reducing response times. However, studies focusing on integrating such systems with mobile applications for remote control by traffic police are sparse.

5. Real-time Traffic Management Systems:
Literature on real-time traffic management systems often focuses on overall traffic flow optimization rather than specific prioritization of emergency vehicles. However, methodologies for real-time traffic monitoring and control, as discussed by Chen et al. (2017), provide valuable insights into the design and implementation of systems capable of promptly responding to changing traffic conditions.
The field of smart ambulance detection and traffic management has witnessed significant advancements in recent years, fueled by the proliferation of image processing techniques and the growing demand for efficient emergency response systems. Studies have explored a range of methodologies for ambulance detection, including traditional computer vision algorithms like Haar cascades and more advanced deep learning approaches such as convolutional neural networks (CNNs). These methods have demonstrated varying degrees of accuracy and computational efficiency in identifying ambulance vehicles amidst complex traffic scenes.

While limited literature specifically addresses the integration of ambulance detection systems with external applications like the Massachusetts Institute of Technology (MIT) app for emergency communication, studies on mobile app integration with IoT devices offer insights into the potential challenges and benefits of such integrations.Furthermore, research on traffic signal control strategies has explored dynamic adjustments based on real-time ambulance detection to prioritize emergency vehicles. However, there's a notable gap in the literature concerning the integration of these systems with mobile applications for remote control by traffic police, highlighting an area ripe for further exploration and development.

The android application has four buttons for four directions. It depends on the route which ambulance driver will select, the ambulance driver will select the appropriate direction and send activate command for the particular signal. In the android application, the patient's information is also stored which consists of name, age, and blood group, besides others. The hardware module for traffic signal has used Arduino for traffic signals. It consists of a Wi-Fi module, with the help of which it captures information from the server and connects the android application directly with the traffic signal. The location is retrieved in the form of longitude and latitude. 5 The direction of ambulance movement depends on the degree of the ambulance. For detecting the direction and current location of the ambulance a compass is used. Methods of traffic surveys in cities for comparison of traffic control systems, Jan Siler, Zuzana Blinova, Martin Langar(1999) felt that traffic overload is a pressing problem for many cities nowadays and they are trying to find smart solutions. Therefore, cities are introducing traffic control systems that facilitate a more dynamic transit through the city for transit traffic while optimizing the control of individual junctions equipped with light signalling. Evaluation of traffic control parameters is usually based on processing of data from traffic surveys. This article describes the design, implementation and verification of alternative methods of traffic surveys for the comparison of traffic control systems in a case study of Huesker Hardest. In this city, an extensive pilot project took place in summer and autumn 2017 to test the new algorithm for adaptive traffic control at selected light-controlled junctions of the city.

The aim of the designed traffic surveys was to evaluate the parameters of the newly installed traffic control systems and to compare them with the control parameters of the original system The hardware used in this system is short-ranged and have low power. Kandhari and Antonov proposed a system, which segments the overall system into three phases, which include counting and detection of vehicles, detection of ambulance and counting and databased decision makings based on data. The system made use of a detection mechanism, which based its decision on images. Therefore, this system does not use electronic sensors. The route is cleared for emergency vehicles stuck in traffic using Bluetooth technology. Bluetooth module and a Bluetooth-enabled phone with Bluetooth state active are required for the detection phase. For this system to work, the ambulance should be near the signal for sending the command for guiding the traffic signal to the Bluetooth module. The range of Bluetooth is very low so there can be a case that the driver sends the command using his phone and it was not received by the Bluetooth module. This system suffers from security-related issues. In order

to make it secure the code needs to be changed every 24 hours. 6 Intelligent traffic control system based single chip microcomputer,Authors of this were Shen Rui Bing ,Hu Shije (2001).This design is based on single chip microcomputer intelligent traffic control system, function complete two main aspects: normal traffic flow, respectively control the north-south and east-west direction of red, green and yellow lights out, When the north-south intersection traffic is big, can increase the north-south intersection of the green light time, when things intersection traffic is big, can increase the intersection of the green light time, back to normal after the end.

The system proposed by Ramani and Jeyakumar makes u of a central server for controlling the traffic controllers. Arduino UNO is used for the implementation of the traffic signal controller. In this system a web application is provided to the ambulance driver. The request for turning the signal light to green is sent to the traffic controller using the web application. The system is divided into three parts namely, web application, cloud server and traffic signal. The ambulance driver uses the web application for choosing the route and navigating the ambulance. Communication between the ambulance and traffic signal is established using the cloud server. Arduino UNO is interfaced with Wi-Fi module and the Wi-Fi module is used as a traffic signal in this system. This system always requires sending of a signal to turn the traffic signal green, as it is not automatic. There could be a case in which the driver forgets to send the signal because of the time constrains. In that case, the system will not work. For this system to work properly, Internet should be available to the drivers all the Tim which is not possible in all areas. Density based smart traffic density based smart traffic control system using canny edge detection algorithm for congregation,Taqi Tahmid, Ekal's Hossain (2011)proposed that as the problem of urban traffic congestion intensifies, there is a pressing need for the introduction of advanced technology and equipment to improve the state-of-the art of traffic control.

The current methods used such as timers or human control are proved to be inferior to alleviate this crisis. In this paper, a system to control the traffic by measuring the real-time vehicle density using canny edge detection with digital image processing is proposed. This imposing traffic 7 control system offers significant improvement in response time, vehicle management, automation, reliability and overall efficiency over the existing systems. Besides that, the complete technique from image acquisition to edge detection and finally green signal allotment using four sample images of different

traffic conditions is illustrated with proper schematics and the final results are verified by hardware implementation. The system used in this paper in manual, neither does it does not give the shortest path to the hospital nor does the signal changes automatically. Parida et al.uses an RF transmitter-receiver module and a Zigbee transmitterreceiver along with At mega 328 IS's. This system enables emergency vehicles to override current traffic light sequence and reach its destination uninterrupted. The whole system depends on communication between emergency vehicles and traffic signals. The RF transmitter-receiver module and a Zigbee transmitterreceiver are both connected to the microcontroller.

There could be a case when there is a traffic jam caused by minor accident or road construction, which will not be known to the emergency vehicle driver. So in that case, this system is of no use to the emergency vehicle driver, as the emergency vehicle will still be stuck in a traffic jam even after having control of traffic light smart traffic system using traffic flow models, Authors of this were Arihant Kamdar, Dr.Jigarkumar Shah discussed that in today's world, one of the biggest issues faced by transportation is related to roadway traffic. While developments are focused on improving the vehicles and making them more efficient, less focus is given to improve the infrastructure and management of the traffic system on intersections like crossroads and solving the problem of traffic Congestion. The existing Systems of the traffic management system at junctions are very inefficient as traffic is static in nature. Static nature means the traffic control system/management system is independent of features that changes with time. There are some systems proposed earlier which makes the system more efficient but they are not efficient as they are not fully dynamic. The system proposed here works on the dynamic behaviour of traffic in order to reduce the traffic congestion which is dependent on real-time factors which are dynamic in nature. Another important feature is priority scheduling for Emergencies/Special Vehicles like Ambulance is also implemented.

This method uses computer vision like vehicle detection and vehicle classification for implementing the same. The aim of the proposed system is to help in the reduction of traffic jamming and make the transportation time-efficient at a cost-effective implementation. Android and cloud based traffic control system, Authors of this were Mpho K. Madisa1, Meera K. Joseph(2004). Traffic congestion has become a major problem in the urban cities, and it is mainly caused due to an increase in population which causes traffic congestion. It therefore becomes a problem for the emergency

vehicles for example, an ambulance, to arrive as early as possible to its destination without appropriate traffic control system. So it was essential to design a traffic control system which will be controlled by an android mobile device via the cloud environment. In this paper we explored other intelligent traffic light systems and other transportation systems. This system has to allow the Android mobile device (emergency vehicle) to override the normal operation of a traffic signal. An efficient and cost-effective system which can solve this problem is an Android and Cloud based traffic control system using the GSM module. The system comprises of five stages which are Android mobile device, GSM module, MQTT (IoT) for Arduino IDE, Arduino Uno microcontroller and traffic signals.

The developed system has allowed the Android mobile device (emergency vehicle) to override the normal operation of a traffic signal. The direction of ambulance movement depends on the degree of the ambulance. For detecting the direction and current location of the ambulance a compass is used. The system used in this in manual, neither does it does not give the shortest path to the hospital nor does the signal changes automatically. Parida et al. uses an RF transmitter-receiver module and a Zigbee transmitter-receiver along with At mega 328 IS's. This system enables emergency vehicles to override current traffic light sequence and reach its destination uninterrupted. The whole system depends on communication between emergency vehicles and traffic signals. The RF transmitter-receiver module and a Zigbee transmitter-receiver are both connected to the microcontroller. There could be a case when there is a traffic jam caused by minor accident or road construction, which will not be known to the emergency vehicle driver. Smart traffic control system using green energy, Authors of this were Jadhav Shruti, Patil Dipali, Gavage Rani, Kulkarni V.A., Patil Amey, Shinde 9 Pradip(2013)discussed that now days the rapidly increasing vehicles & large time delays between traffic lights leads to wastage of fuel and time .To overcome this problem, we go through smart traffic control system using green energy.

The conventional traffic lights are works on fixed time period allotted to each side of junction which cannot be varied & junction timing allotted are fixed In the smart traffic control system the traffic is controlled according to traffic density within the particular distance. This system is done using Arduino mega interfaced with IR sensors & timing at the junction changes automatically. This system comes under digital India campaign and supports Smart City Concept. So in that case, this system is

of no use to the emergency vehicle driver, as the emergency vehicle will still be stuck in a traffic jam even after having control of traffic light. The route is cleared for emergency vehicles stuck in traffic using Bluetooth technology. Bluetooth module and a Bluetooth-enabled phone with Bluetooth state active are required for the detection phase. For this system to work, the ambulance should be near the signal for sending the command for guiding the traffic signal to the Bluetooth module. The range of Bluetooth is very low so there can be a case that the driver sends the command using his phone and it was not received by the Bluetooth module. This system suffers from security related issues. In order to make it secure the code needs to be changed every 24 hours.

The system proposed by Ramani and Jeyakumar makes use of a central server for controlling the traffic controllers. Intelligent traffic control system based on a single chip microcomputer, Authors of this were Hu ZhiJie,Shen Rui Bing (2017)proposed that design is based on single chip microcomputer intelligent traffic control system, function complete two main aspects: normal traffic flow, respectively control the north-south and east-west direction of red, green and yellow lights out; When the north-south intersection traffic is big, can increase the north south intersection of the green light time, when things 10 intersection traffic is big, can increase the intersection of the green light time, back to normal after the end. This design meets the intelligent system, reliability and real-time requirements Arduino UNO is used for the implementation of the traffic signal controller. In this system a web application is provided to the ambulance driver. The request for turning the signal light green is sent to the traffic controller using the web application. The system is divided into three parts namely, web application, cloud server and traffic signal. The ambulance driver uses the web application for choosing the route and navigating the ambulance. Communication between the ambulance and traffic signal is established using the cloud server. Arduino UNO is interfaced with Wi-Fi module and the Wi-Fi module is used as a traffic signal in this system. This system always requires sending of a signal to turn the traffic signal green, as it is not automatic.

Kandhari and Antonov proposed a system, which segments the overall system into three phases, which include counting and detection of vehicles, detection of ambulance and counting and data-based decision makings based on data. The system made use of a detection mechanism, which based its decision on images. Therefore, this system does not use electronic sensors. Priority based real time smart traffic control system using dynamic background, Sarath S and Deepthi L.R. proposed(2001)

that vehicular traffic is increasing rapidly in this world which is resulting in traffic congestion. The emergency vehicles such as ambulances, fire engines and police vehicles are given equal priority over other vehicles and hence get stuck up in this traffic congestion. A methodology for priority-based vehicle detection based on image processing techniques is proposed in this paper. If an emergency vehicle is detected on the road, the lane in which this vehicle is will be given higher priority over all other lanes. The operation of these elements is described below. The android application has four buttons for four directions. It depends on the route which ambulance driver will select, the ambulance driver will select the appropriate direction and send activate command for the particular signal. In the android application, the patient's information is also stored which consists of name, age, and blood group, besides others. The hardware module for traffic signal has used Arduino for traffic signals. It consists of a Wi-Fi module, with the help of which it captures information from the server and connects the android application directly with the traffic signal. The location is retrieved in the form of longitude and latitude. The direction of ambulance movement depends on the degree of the ambulance. For detecting the direction and current location of the ambulance a compass is used The system used in this paper in manual, neither does it does not give the shortest path to the hospital nor does the signal changes automatically. Parida et al. uses an RF transmitter-receiver module and a Zigbee transmitter-receiver along with At mega 328 IS's. This system enables emergency vehicles to override current traffic light sequence and reach its destination uninterrupted. The whole system depends on communication between emergency vehicles and traffic signals.

There could be a case when there is a traffic jam caused by minor accident or road construction, which will not be known to the emergency vehicle driver. So in that case, this system is of no use to the emergency vehicle driver, as the emergency vehicle will still be stuck in a traffic jam even after having control of traffic light. The hardware used in this system is short-ranged and have low power. The system made use of a detection mechanism, which based its decision on images. Therefore, this system does not use electronic sensors. The route is cleared for emergency vehicles stuck in traffic using Bluetooth technology. Bluetooth module and a Bluetooth-enabled phone with Bluetooth state active are required for the detection phase. For this system to work, the ambulance should be near the signal for sending the command for guiding the 13 traffic signal to the Bluetooth module. The range of Bluetooth is very low so there can be a case that the driver sends the command using his phone and it was not received by the Bluetooth module. This system suffers from securityrelated issues. In

order to make it secure the code needs to be changed every 24 hours. The system proposed by Ramani and Jeyakumar makes use of a central server for controlling the traffic controllers. Arduino UNO is used for the implementation of the traffic signal controller. In this system, a web application is provided to the driver. The request for turning the signal light to green is sent to the traffic controller using the web application. The system is divided into three parts namely web application cloud server and traffic signal. The ambulance driver uses the web application for choosing the route and navigating the ambulance. Communication between the ambulance and traffic signal is established using the cloud server. Arduino UNO is interfaced with Wi-Fi module and the Wi-Fi module is used as a traffic signal in this system. This system always requires sending of a signal to turn the traffic signal green, as it is not automatic. There could be a case in which the driver forgets to send the signal because of the time constrains.

1.5 PROBLEM STATEMENT

The problem addressed in this research is the need for an integrated system for smart ambulance detection and traffic management to optimize emergency response efficiency in urban environments. Current methods for ambulance detection often lack real-time capabilities and fail to consider varying distances between ambulances and the camera. Additionally, there is a lack of systems that seamlessly integrate with mobile applications for emergency communication and traffic signal control by authorities.

The challenges include developing robust image processing algorithms capable of accurately detecting ambulances in diverse traffic scenarios, implementing efficient distance estimation techniques to prioritize emergency vehicles based on proximity, and seamlessly integrating the detection system with external applications like the Massachusetts Institute of Technology (MIT) app for emergency communication and traffic signal control. Furthermore, ensuring the reliability, scalability, and real-time performance of the system poses additional technical hurdles.

Addressing these challenges requires a multidisciplinary approach, incorporating expertise in computer vision, machine learning, real-time systems, mobile application development, and traffic engineering. The proposed solution aims to create a comprehensive framework that not only detects

ambulances in real-time but also facilitates effective communication between emergency responders and traffic authorities to expedite emergency response and ensure smooth traffic flow.

1.6 OBJECTIVES

Case 1: Remote Emergency Notification by Ambulance Driver

Develop a communication system integrated with the MIT app that allows ambulance drivers to notify traffic authorities of an emergency situation when the ambulance is far from the camera's detection range. This system should enable drivers to send an "emergency" message through the app, which will then be displayed on an LCD screen visible to traffic authorities and other relevant personnel. This ensures that even when the ambulance is not in the immediate vicinity of the camera, traffic control can be alerted to prepare for the incoming emergency vehicle, thereby enabling a proactive response to facilitate its passage.

Case 2: Manual Traffic Signal Control by Traffic Police

Implement a feature in the Arduino Bluetooth app that allows traffic police to manually change the traffic signal LEDs remotely. When the LED is changed to green, the system should trigger a buzzer sound to alert nearby personnel and drivers that the signal has been modified to prioritize the emergency vehicle's route. This manual control mechanism provides a reliable fallback option for traffic authorities to ensure ambulances can pass through intersections smoothly, especially in scenarios where automatic detection systems may fail or need reinforcement.

Case 3: Automatic Traffic Signal Control Based on Ambulance Detection

Design and deploy a robust image processing algorithm capable of real-time ambulance detection using camera feeds. Upon detecting an ambulance, the system should automatically change the traffic signal LED to green, ensuring a clear path for the emergency vehicle. This automated control minimizes human intervention, reduces response time, and enhances the efficiency of emergency vehicle prioritization. The objective is to create a seamless, reliable system that dynamically adjusts traffic signals based on real-time data, thus improving the overall effectiveness of emergency response and traffic management.

The proposed Smart Ambulance Detection and Traffic Management System aims to revolutionize emergency response and traffic management in urban environments. By leveraging advanced image processing algorithms and integrating cutting-edge technologies like the MIT app, the system ensures real-time detection and prioritization of ambulances. It provides a robust communication channel for ambulance drivers, enabling early emergency alerts and facilitating proactive traffic signal adjustments by traffic authorities. The automated signal control based on real-time ambulance detection minimizes delays and enhances the efficiency of emergency routes. Moreover, the system's design emphasizes scalability, interoperability, and data security, ensuring reliable and seamless operation within existing traffic infrastructure. By reducing response times, improving air quality, and enhancing public safety, this innovative approach not only addresses immediate logistical challenges but also contributes to the broader societal goal of creating safer and more efficient urban spaces.

CHAPTER 2

SYSTEM ANALYSIS

2.1 TRADITIONAL METHODS OF DETECTING AMBULANCE

2.1.1 Color based Detection:

Ambulances typically have a distinctive color scheme, often featuring a combination of white or light yellow with bright red or orange stripes. Additionally, emergency lights on the vehicle emit specific colors like red or blue. Color based detection algorithms leverage these characteristics by segmenting the image based on color information.Techniques such as thresholding, where pixels with color values falling within predefined ranges are classified as belonging to an ambulance, or clustering algorithms like kmeans clustering, which group pixels with similar colors together, are commonly employed.While effective in scenarios with consistent lighting conditions and unobstructed views of the ambulance, colorbased methods may struggle in situations with varying lighting, shadows, or when other objects with similar colors are present in the scene.

2.1.2 Template Matching:

Template matching involves comparing a predefined template image of an ambulance with regions of the input image to find areas that closely match the template. Common metrics for comparison include normalized crosscorrelation or sum of squared differences.While straightforward to implement, template matching may be sensitive to variations in scale, orientation, and viewpoint. Additionally, it can be computationally expensive when searching over large regions of the image at multiple scales.

2.1.3 Edge Detection:

Edge detection algorithms, such as the Canny edge detector, identify sharp changes in intensity in the image, which often correspond to object boundaries.By detecting edges and applying criteria such as size, shape, and symmetry, regions likely to contain ambulances can be extracted from the edge map. Additional processing steps, such as contour analysis or region growing, may be employed to refine the detection.

2.1.4 Haar Cascade Classifiers:

Haar cascade classifiers are machine learningbased methods that use a set of predefined features to detect objects in images. These features are trained on positive and negative examples of the target object, allowing the classifier to learn discriminative patterns.While originally designed for face detection, Haar cascades can be adapted to recognize the shapes and patterns characteristic of ambulances, such as the boxy shape and prominent emergency lights.

2.1.5 Motion Detection:

Motion detection algorithms exploit the fact that ambulances are often in motion. By comparing successive frames of a video feed, changes in pixel values caused by moving objects can be identified.Techniques like background subtraction, where the stationary background is subtracted from each frame to isolate moving objects, or frame differencing, which calculates the absolute difference between consecutive frames, are commonly used.Motion detection can be particularly effective in scenarios where ambulances are moving against a relatively static background, such as on highways or urban streets.

2.2 MODERN PRACTICES

2.2.1 Convolutional Neural Networks (CNNs):

CNNs have revolutionized object detection tasks, including ambulance detection. These deep learning architectures are composed of multiple layers of convolutional, pooling, and fully connected layers, allowing them to automatically learn hierarchical representations of visual data. During training, the CNN learns to identify discriminative features and patterns that distinguish ambulances from other objects in the scene. Architectures such as Faster R-CNN, YOLO (You Only Look Once), and SSD (Single Shot Multibox Detector) are commonly used for this purpose.

2.2.2 Data Augmentation:

Data augmentation techniques are employed to artificially increase the diversity of the training data and improve the robustness of CNN models to variations in lighting, viewpoint, and occlusions. Common data augmentation techniques include random rotations, translations, flips, changes in brightness or contrast, and adding noise to the images. By augmenting the training data with these

transformations, the CNN learns to generalize better to unseen variations in the test data, resulting in more robust ambulance detection models.

2.2.3 Ensemble Methods:

Ensemble methods combine predictions from multiple CNN models trained with different architectures or on different subsets of the training data. By aggregating the outputs of diverse models, ensemble methods can improve overall detection accuracy and generalization performance. Techniques such as model averaging, stacking, and boosting are commonly used to create ensembles of CNN models for ambulance detection. Ensemble methods are particularly effective when individual models exhibit complementary strengths and weaknesses, leading to improved detection performance when combined.

2.2.4 Context Modeling:

Context modeling techniques consider the spatial relationships and contextual cues surrounding the ambulance in the image. By incorporating contextual information such as road layouts, traffic patterns, and the presence of other vehicles or pedestrians, context-aware models can make more informed decisions about ambulance detection. Context modeling helps the model understand the scene context and distinguish between true ambulance sightings and false positives caused by similar-looking objects or environmental factors.

2.2.5 Real-time Processing:

Real-time processing is essential for ambulance detection systems deployed in traffic management and emergency response applications, where timely detection and response are critical. Efforts are made to optimize CNN architectures and implement efficient inference algorithms to enable real-time processing of video feeds from traffic cameras. Techniques such as model quantization, network pruning, and hardware acceleration using specialized processing units (e.g., GPUs, TPUs) are employed to reduce inference latency and enable real-time performance. By achieving real-time processing capabilities, ambulance detection systems can facilitate smoother traffic flow, quicker emergency response times, and enhanced public safety outcomes.

2.3 OBJECT DETECTION

Object detection is a computer vision task that involves identifying and localizing objects within an image or video frame. Unlike image classification, which only determines the presence of objects

in an image, object detection provides information about the location of each object by drawing bounding boxes around them.

This task is fundamental to a wide range of applications, including autonomous driving, surveillance, augmented reality, and medical imaging. Various approaches to object detection exist, ranging from traditional methods like sliding window techniques and feature-based classifiers to more recent deep learning-based approaches such as convolutional neural networks (CNNs) and their derivatives like Region-based Convolutional Neural Networks (R-CNN), You Only Look Once (YOLO), and SSD. Deep learning-based methods have gained prominence due to their ability to learn hierarchical representations of visual data directly from raw pixel inputs, resulting in superior accuracy and efficiency in detecting objects across diverse contexts and environments. Object detection systems typically comprise multiple stages, including feature extraction, region proposal generation, object classification, and bounding box regression. These systems are trained on large-scale annotated datasets to learn discriminative features and optimize detection performance. As object detection continues to advance, with the integration of cutting-edge techniques such as attention mechanisms and context modeling, its capabilities are poised to revolutionize numerous industries and contribute to the development of intelligent systems with enhanced perception and understanding of the visual world.

2.4 EMERGENCY VEHICLE DETECTION

Emergency vehicle detection is a specialized application of object detection within the domain of computer vision, focusing specifically on identifying vehicles used for emergency services such as ambulances, fire trucks, and police cars. This task is crucial for optimizing emergency response systems, as it enables the rapid identification and prioritization of emergency vehicles in traffic scenarios. Emergency vehicle detection systems utilize various techniques, ranging from traditional image processing algorithms to state-of-the-art deep learning methods. These techniques often leverage features unique to emergency vehicles, such as flashing lights, sirens, or distinct color patterns, to differentiate them from other vehicles on the road. In recent years, deep learning approaches, particularly convolutional neural networks (CNNs), have demonstrated significant advancements in emergency vehicle detection due to their ability to automatically learn discriminative features from large-scale datasets. These systems typically involve preprocessing steps such as image enhancement and feature extraction, followed by classification of candidate regions as

emergency vehicles or background. Real-time performance is often a critical requirement for such systems, necessitating efficient algorithms and hardware implementations to ensure timely detection in dynamic traffic environments. Successful deployment of emergency vehicle detection systems can lead to reduced response times for emergency services, improved traffic flow management, and enhanced safety for both emergency responders and the general public. Ongoing research in this field continues to refine and innovate detection techniques, with the ultimate goal of creating robust and reliable systems capable of seamlessly integrating into existing emergency response infrastructure.

2.4 Ambulance Detection

2.5 USE CASE DIAGRAM

A use case is a diagram showing the various user roles and the way those users interact with the system and at its simplest it is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case is a methodology used in system analysis to identify, clarify and organize system requirements. In this context, the term system refers to something being developed or operated. Use case diagrams are employed in UML, a standard notation for the modelling of real-world objects and systems. It can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. It is represented by either circles or ellipses. The boundary, which defines the system of interest in relation to the world around it. The actors, usually individuals involved with the system defined according to their roles. The use cases, in which the specific roles are played by the actors within and around the system. The relationships between and among the actors and the use cases. The use case diagram looks something like a flowchart. Intuitive symbols represent the system elements. The use case diagrams serve as a kind of table of contents for the data activities that must be supported by the system. It is used to identify the uses or use cases of

the new system-in other words, to identify how the system will be used and which actors will be involved in which use cases. A use case diagram is a convenient way to document the system events. Sometimes a single, comprehensive diagram is used to identify all use cases for an entire system. At other times, a set of narrower use case diagrams is used.

2.6 SEQUENCE DIAGRAM

Sequence diagram is an interaction diagram that shows how processes operate with one another and what is their order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios. A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

CHAPTER 3

SYSTEM REQUIREMENT SPECIFICATION

3.1 INTRODUCTION TO SPECS

Framework Requirement Specification (SRS) is a focal report, which outlines the foundation of the item headway handle. System Requirement Specification (SRS) is a central report, which frames the establishment of the product advancement process. It records the necessities of a structure and in addition has a delineation of its noteworthy highlight. An SRS is basically an affiliation's seeing (in making) of a customer or potential client's edge work necessities and conditions at a particular point in time (for the most part) before any veritable design or change work. It's a two-way insurance approach that ensures that both the client and the affiliation understand exchange's necessities from that perspective at a given point in time. The synthesis of programming need detail reduces headway effort, as careful review of the report can reveal oversights, mixed up presumptions, and inconsistencies in front of plan for the change cycle when these issues are less requesting to right. The SRS discusses the thing however not the wander that made it, thus the SRS fills in as a start for later change of the finished thing. The SRS may should be changed, be that as it may it gives a foundation to continue with creation appraisal. In direct words, programming need assurance is the starting phase of the item change activity. The SRS implies unraveling the musings in the brains of the clients the data, into a formal chronicle the yield of the essential stage.

3.1.1 Estimating Hardware

1. Arduino Uno
2. HC-05 Bluetooth module
3. Liquid Crystal Display
4. ESP8266 WiFi Module
5. Webcam
5. Power Supply Unit
6. Jumper Wires

3.1.2 Hardware Functionality:

1. Arduino Uno

General elucidation:

The Arduino Uno is a highly popular microcontroller board designed for a wide range of electronics projects, from simple to complex. It is based on the ATmega328P microcontroller, which is an 8-bit chip with 32KB of flash memory, 2KB of SRAM, and 1KB of EEPROM. This provides ample storage for code and data for a variety of applications.The board features 14 digital input/output pins, numbered from 0 to 13. These pins can be used to interface with other electronic components, allowing the Arduino to read digital signals (either high or low, corresponding to 5V or 0V) and control devices such as LEDs, motors, or sensors. Additionally, 6 of these digital pins (specifically pins 3, 5, 6, 9, 10, and 11) can produce Pulse Width Modulation (PWM) signals. PWM is a technique for simulating an analog output by varying the duty cycle of a digital signal, which is useful for controlling the brightness of LEDs or the speed of motors.

Moreover, the Arduino Uno includes 6 analog input pins (labeled A0 to A5). These pins can read analog signals from sensors, allowing the board to interpret varying voltage levels between 0V and 5V. This is crucial for applications that require measuring environmental conditions like temperature, light intensity, or pressure.The board operates at 5V and can be powered either through a USB connection or an external power supply, with an input voltage range of 7-12V for the latter. It also features a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP (In-Circuit Serial Programming) header, and a reset button. The USB connection not only provides power but also enables communication with a computer, allowing for easy programming and interaction through the Arduino Integrated Development Environment (IDE).

3.1.2.1 Arduino Uno

3.1.2.1 *Pin configuration of Arduino UNO :*

SNO	PINS	DEFINITION OF PINS
1	Digital Pins (D0-D13)	These pins can be used for digital input or output. They operate at 5V and can be configured as either input (reading digital signals) or output (sending digital signals).
2	Analog Pins (A0-A5)	These pins are used for analog input. They can read analog voltages (0-5V) from sensors or other devices.
3	Power Pins	- 5V: Provides a regulated 5V power supply for external components. - 3.3V: Provides a 3.3V power supply. - GND (Ground): Connects to the ground reference.
4	Reset Pin (RESET)	Used to reset the microcontroller. Pulling this pin LOW resets the board.
5	TX/RX Pins	- TX (Transmit): Sends serial data from the board. - RX (Receive): Receives serial data into the board.
6	Crystal Oscillator Pins (XTAL1, XTAL2)	Connect to an external crystal oscillator for accurate timing.
7	Voltage Regulator	Regulates the input voltage (usually 7-12V) to 5V for the board.
8	USB Connector	Used for programming and communication with the computer.
9	ICSP Header	In-Circuit Serial Programming header for advanced programming.
10	LEDs (Power, L)	- Power LED: Indicates that the board is powered. - L LED: Connected to digital pin 13; can be used for debugging.

Applications:

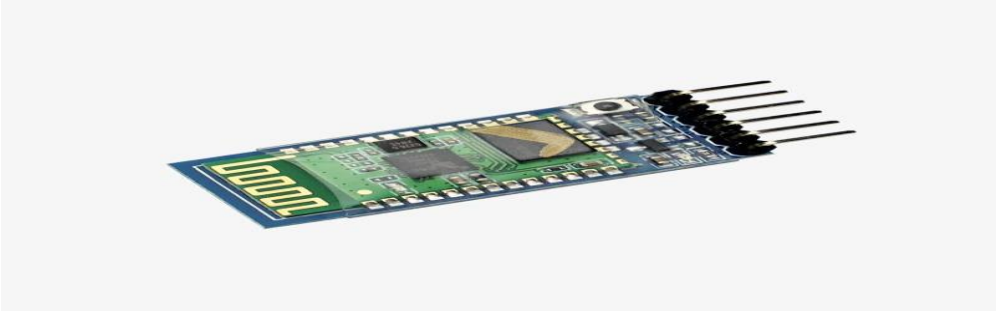
1. Home Automation
2. Robotics
3. Wearable Technology
4. Environmental Monitoring
5. Interactive Art Installations

2. HC-05 Bluetooth module

General elucidation:

The HC-05 Bluetooth module is a widely used Bluetooth-to-serial interface module designed for wireless communication in embedded systems and microcontroller projects. It allows devices to communicate with each other wirelessly over a short range, typically up to 10 meters. The HC-05 can operate in either master or slave mode, making it versatile for various applications where a reliable, low-cost Bluetooth connection is needed.The module uses Bluetooth 2.0 + EDR (Enhanced Data Rate) technology, providing data transfer rates of up to 3 Mbps. Despite this, it is commonly connected to 5V systems like the Arduino with proper level shifting or resistor voltage dividers.

The HC-05 has a straightforward interface, including key pins for power (VCC and GND), and serial communication (TXD and RXD). It also has a state pin (STATE) that indicates the module's status and a key pin (EN) used to switch between command and data modes.In command mode, the HC-05 can be configured using AT commands, which allow users to change settings like the baud rate, device name, pairing password, and more. In data mode, the module transmits and receives data wirelessly, which can then be read or sent by a connected microcontroller.



3.1.2.2 Bluetooth Module

3.1.2.2 *Pin configuration of Bluetooth Module :*

SNO	PINS	DEFINITION OF PINS
1	Enable/Key	This pin toggles between Data Mode (set low) and AT command mode (set high). By default, it is in Data mode.
2	Vcc	Powers the module. Connect to +5V Supply voltage.
3	Ground	Ground pin of the module, connect to system ground.
4	TX	Transmitter pin that transmits Serial Data. Everything received via Bluetooth will be given out by this pin as serial data.
5	RX	Receiver pin that receives Serial Data. Every serial data given to this pin will be broadcasted via Bluetooth.
6	State	The state pin is connected to an onboard LED; it can be used as feedback to check if Bluetooth is working properly.
7	LED	Indicates the status of the Module. Blink patterns: Once in 2 sec for Command Mode, Repeated Blinking for waiting connection in Data Mode, Twice in 1 sec for Connection successful in Data Mode.
8	Button	Used to control the Key/Enable pin to toggle between Data and Command Mode.

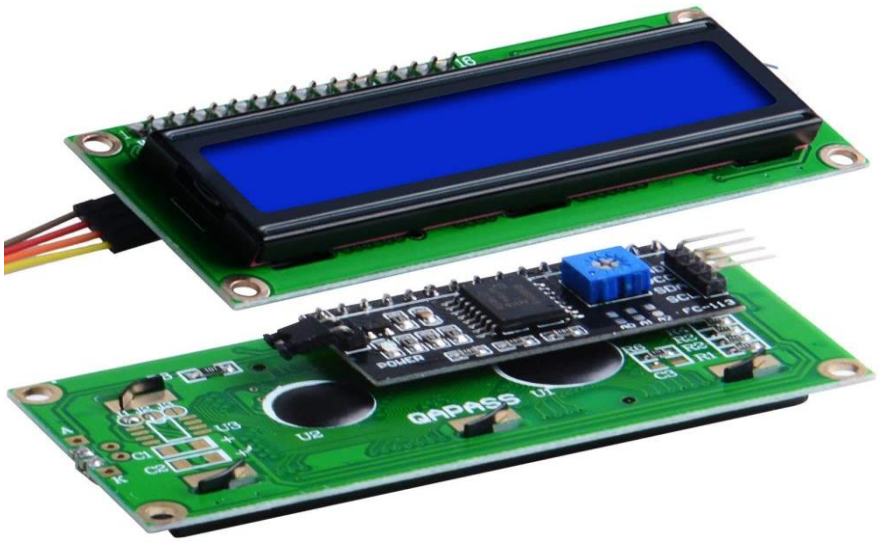
Applications:

1. Wireless Data Transfer
2. Remote Control Systems
3. Home Automation
4. Wireless Sensors Networks
5. Health Monitoring System

3. Liquid Crystal Display

General elucidation:

The I2C (Inter-Integrated Circuit) Liquid Crystal Display (LCD) is a type of LCD screen that utilizes the I2C communication protocol for interfacing with microcontrollers or other devices. Unlike traditional parallel LCD displays that require a significant number of pins for communication, I2C LCDs only require two pins (SDA and SCL) to transmit data and commands, making them ideal for projects with limited available pins or for simplifying wiring complexity.



3.1.2.3 LCD Display

These displays typically consist of a monochrome or color LCD screen, a controller chip, and an I2C interface module. The controller chip manages the display's operation and can handle tasks such as refreshing the screen content, interpreting commands, and controlling the backlight. The I2C interface

module acts as a bridge between the LCD and the microcontroller, facilitating communication between the two using the I2C protocol.One of the key advantages of using I2C LCDs is their ease of use and versatility. They can be easily integrated into various projects, including embedded systems, IoT devices, robotics, and consumer electronics. Additionally, the I2C protocol allows multiple devices to share the same communication bus, enabling efficient communication between multiple peripherals without the need for additional pins.Overall, I2C LCDs offer a convenient and efficient solution for displaying information in embedded systems and other projects, providing a compact, low-power, and easy-to-use interface for visual feedback and user interaction.

3.1.2.3 *Pin configuration of LCD display:*

SNO	**PINS**	**DEFINITION OF PINS**
1	**GND**	Ground pin. Connect to system ground (0V).
2	**VCC**	Power supply pin. Connect to +5V output of the Arduino or an external 5V power supply.
3	**SDA**	I2C data pin. Used for communication between the Arduino and the LCD display.
4	**SCL**	I2C clock pin. Also used for communication between the Arduino and the LCD display.

Applications:

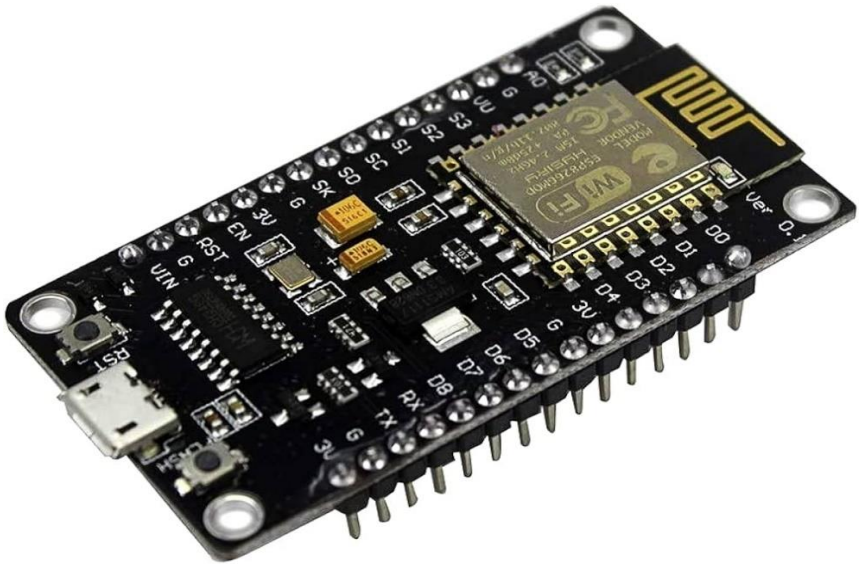
1. User Interfaces
2. DIY Electronics
3. Industrial Testing Tools

4. Character Displays
5. Educational Projects

4. ESP8266 WiFi Module

General elucidation:

The ESP8266 WiFi Module stands as a groundbreaking component in the realm of electronics, renowned for its capability to bestow wireless connectivity upon an array of devices. It embodies a potent fusion of a low-power microcontroller and a WiFi transceiver, affording devices the ability to seamlessly connect to WiFi networks and engage in communication over the internet. This module's compact size and user-friendly nature have propelled it into the limelight across a spectrum of domains, captivating hobbyists, professionals, and developers alike. Its applications span a broad spectrum, ranging from home automation endeavors to Internet of Things (IoT) projects, and from remote monitoring systems to smart agriculture solutions. Thanks to its capacity to integrate with existing WiFi networks and its support for TCP/IP protocols, it serves as a cornerstone for an extensive array of projects necessitating wireless connectivity. Furthermore, its programmability through diverse development environments such as Arduino IDE and MicroPython renders it accessible to individuals of varying proficiency levels in programming. In essence, the ESP8266 WiFi Module empowers creators to forge innovative and interconnected solutions, propelling us further into a realm of connectivity and possibility.



3.1.2.4 ESP8266 WiFi Module

3.1.2.4 *Pin configuration of ESP8266 WiFi Module:*

SNO	PINS	DEFINITION OF PINS
1	**GND**	Ground Pin. Connect to the ground of the circuit.
2	**TX**	Serial Transmit Pin of UART. Connected to Rx pin of programmer/microcontroller to upload program. Can act as a GPIO when not used as TX.
3	**GPIO-2**	General purpose Input/output pin.
4	**CH_EN**	Chip Enable – Active high.
5	**GPIO-0**	Flash General purpose Input/output pin. Takes module into serial programming when held low during startup.
6	**Reset**	Resets the module.
7	**RX**	Serial Receive Pin of UART. General purpose Input/output pin. Can act as a GPIO when not used as RX.
8	**Vcc**	Connect to +3.3V only.

Applications:

1. IoT (Internet of Things) Devices
2. Wearable Technology
3. Smart Agriculture
4. Industrial Monitoring and Control
5. Environmental Monitoring
6. Remote Sensing

5. Webcam

A webcam is a digital camera device typically attached to a computer or network, used to capture and transmit live video footage or still images in real-time. It consists of a lens, an image sensor, and electronics for processing and transmitting the captured video or images. Webcams are commonly used for various purposes, including video conferencing, live streaming, video surveillance, online education, and social media interaction. They enable users to communicate visually over the internet, allowing for face-to-face conversations and virtual meetings regardless of geographical location. Webcams come in various forms, including built-in webcams integrated into laptops and external webcams connected to desktop computers via USB or other interfaces. They have become essential tools for remote work, online learning, and staying connected with friends and family, especially during times when physical gatherings are limited. Additionally, webcams are often used in security systems for monitoring and recording activities in homes, offices, and public spaces.

3.1.2.5 Webcam

5. Power Supply

Unit Used to power the components.

6. Jumper Wires

Jumper wires are short electrical wires with connector pins at each end, designed for use with breadboards and other prototyping tools to create temporary circuits without soldering. They come in various colors for easy identification and organization of different connections within a circuit. Jumper wires are typically categorized by their connector types, which can be male-to-male, male-to-female, or female-to-female, allowing them to connect various types of components and devices. These wires are essential tools in electronics prototyping, enabling quick and flexible connections between components such as resistors, capacitors, integrated circuits, and microcontrollers. They are commonly used in educational settings, maker projects, and by hobbyists and professionals alike to experiment with circuit designs, test components, and troubleshoot issues. The use of jumper wires facilitates a modular approach to circuit building, making it easy to modify and expand circuits without permanent alterations.

3.1.2.6 Jumper Wires

3.2 SOFTWARE REQUIREMENTS:

3.2.1 Arduino IDE

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell

your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

3.2.2 Embedded C

Embedded C is a specialized version of the C programming language designed for programming embedded systems. Embedded systems are dedicated computer systems that perform specific functions within larger mechanical or electrical systems, often with real-time computing constraints. Embedded C extends standard C by providing additional libraries and features tailored to the needs of embedded programming, such as direct access to hardware, real-time operation, and efficient use of resources. It enables developers to write code that can interact directly with the hardware components of microcontrollers, sensors, and other embedded devices.

Embedded C is widely used due to its efficiency, portability, and the control it offers over hardware. It allows for precise manipulation of memory and processor operations, which is critical in environments with limited resources. Typical applications include automotive systems, home appliances, medical devices, industrial automation, consumer electronics, and telecommunications equipment. The language's syntax and semantics are similar to standard C, making it accessible to C programmers, but it also requires a good understanding of the underlying hardware to manage tasks like interrupt handling, timing, and power management effectively. The use of Embedded C ensures that embedded systems can perform their functions reliably and efficiently, often under strict performance and resource constraints.

3.2.3 Pycharm

PyCharm is an integrated development environment (IDE) specifically designed for programming in Python. Developed by JetBrains, PyCharm offers a comprehensive suite of tools and features that streamline the coding process and enhance productivity. It includes a powerful code editor with intelligent code completion, real-time error detection, and robust refactoring capabilities. PyCharm also provides advanced debugging tools, integrated unit testing, and support for web development frameworks such as Django and Flask. With built-in version control integration, developers can easily manage code changes and collaborate with team members using systems like Git. Additionally, PyCharm supports various plugins and customization options, allowing users to tailor the environment to their specific needs. Its user-friendly interface, combined with its powerful features,

makes PyCharm a popular choice among both novice and experienced Python developers for building, testing, and maintaining their applications.

3.2.4 OpenCV

OpenCV (Open Source Computer Vision Library) is an open-source software library designed for computer vision and machine learning applications. It provides a vast collection of tools and functions for tasks such as image processing, video analysis, and object detection. Developed initially by Intel, OpenCV is now maintained by an active community of developers and contributors. The library is written in C++ and supports multiple programming languages, including Python, Java, and MATLAB, making it accessible to a wide range of users. OpenCV's capabilities include image filtering, edge detection, face recognition, motion tracking, and 3D reconstruction, among others. It is widely used in various fields such as robotics, surveillance, automotive industry, medical imaging, and augmented reality. Its efficiency, flexibility, and extensive documentation make it a popular choice for both beginners and professionals looking to implement computer vision solutions in their projects.

3.2.5 MIT App

MIT App Inventor is a web-based visual programming environment that allows users to create fully functional Android applications without needing extensive coding knowledge. Developed by the Massachusetts Institute of Technology (MIT), this tool uses a drag-and-drop interface, where users can assemble program blocks to create their applications. Each block represents different functions and actions, simplifying the programming process and making it accessible to people of all ages and backgrounds. MIT App Inventor is especially popular in educational settings, where it is used to teach students the fundamentals of programming and app development. It empowers users to build a wide range of applications, from simple games and interactive stories to more complex apps involving sensors, GPS, and data storage. By providing an intuitive and user-friendly platform, MIT App Inventor fosters creativity and innovation, enabling users to bring their ideas to life on Android devices.

CHAPTER 4

SYSTEM DESIGN AND IMPLEMENTATION

System Design is the process of defining the architecture, components, modules, interfaces and data for a system to satisfy specified requirements. System design could be seen as the application of the systems theory to product development. The system design process builds up general framework building design. Programming outline includes speaking to the product framework works in a shape that may be changed into one or more projects. The prerequisite indicated by the end client must be put in a systematically manner. Outline is an inventive procedure; a great configuration is the way to viable framework. The framework "Outline" is characterized as "The procedure of applying different systems and standards with the end goal of characterizing a procedure or a framework in adequate point of interest to allow its physical acknowledgment". Different configuration components are taken after to add to the framework. The configuration detail portrays the components of the framework, the segments or components of the framework and their appearance to end-users. System designing in terms of software engineering has its own value and importance in the system development process as a whole. To mention it may though seem as simple as anything or simply the design of systems, but in a broader sense it implies a systematic and rigorous approach to design such as system which fulfills all the practical aspects including flexibility, efficiency and security.

4.1 IMPLEMENTATION

Implementation is the phase of the undertaking when the hypothetical configuration is transformed out into a working framework. Implementation should be perfect mapping of the design document in a suitable programming language in order to achieve the necessary final product. Often the product is ruined due to incorrect programming language chosen for implementation or unsuitable method of programming. It is better for the coding phase to be directly linked to the design phase in the sense if the design is in terms of object oriented terms then implementation should be preferably carried out in an object oriented way.

IMPLEMENTING HOG FEATURE DESCRIPTOR

In computer vision there are many algorithms that are designed to extract spatial features to identify objects using information about image gradients. HOG, or Histogram of Oriented Gradients, is one of these algorithms. A histogram is an approximate representation of the distribution of numerical data that looks like a looks a bar graph. Each bar represents a group of data that falls in a certain range of values, also called bins. Orientation means the direction or orientation of an image gradient. HOG will produce a histogram of gradient directions in an image.

The HOG algorithm is applied in the following steps:

1. Calculate the magnitude and direction of the gradients at each pixel of the input image. The above figure shows the gradients of a 8*8 cell in the image. The gradient change in the image is represented by a vector at each pixel. The direction of the vectors indicates the direction of the change of pixel intensity, and the magnitude tells us how strong the change in intensity is.
2. Divide the image into cells of the same size .The cells' size is an optional parameter. The size should be chosen in a way that the scale of features will fit to the cell.
3. Group the gradient directions of all pixels in each cell into a specified number of orientation bins. Sum the magnitudes of the gradients in each bin, which will be the heights of the bins. The number of bins is usually set to 9. So that each bin's width will be 20 degrees.
4. Group the cells into blocks of same size (the red sliding window in the below animation). The amount of movement of the block window over the image is called stride. It is usually set to half the block size. The number of cells in the block and the stride are free parameters which set by the user.
5. Normalize the cell histogram according to the other cells in the block. All the normalized histograms from all the blocks will be added up into a single feature vector. This feature vector is called the HOG descriptor.

4.2 DESIGN CONTEMPLATION

The reason for the design is to arrange the arrangement of the issue determined by the necessities report. This stage is the initial phase in moving from issue to the arrangement space. As such, beginning with what is obliged; outline takes us to work towards how to full fill those needs. The configuration of the framework is maybe the most basic component influencing the nature of the product and has a noteworthy effect on the later stages, especially testing and upkeep. Framework

outline depicts all the significant information structure, document arrangement, yield and real modules in the framework and their Specification is chosen. The preliminary design, or design consideration includes, often bridges a gap between design conception and detailed design, particularly in cases where the level of conceptualization achieved during ideation is not sufficient for full evaluation. This stage is the initial phase in moving from issue to the arrangement space.

4.3 SYSTEM ARCHITECTURE

The architectural configuration procedure is concerned with building up a fundamental basic system for a framework. It includes recognizing the real parts of the framework and interchanges between these segments. The beginning configuration procedure of recognizing these subsystems and building up a structure for subsystem control and correspondence is called construction modeling outline and the yield of this outline procedure is a portrayal of the product structural planning. The proposed architecture for this system is given below. It shows the way this system is designed and brief working of the system.

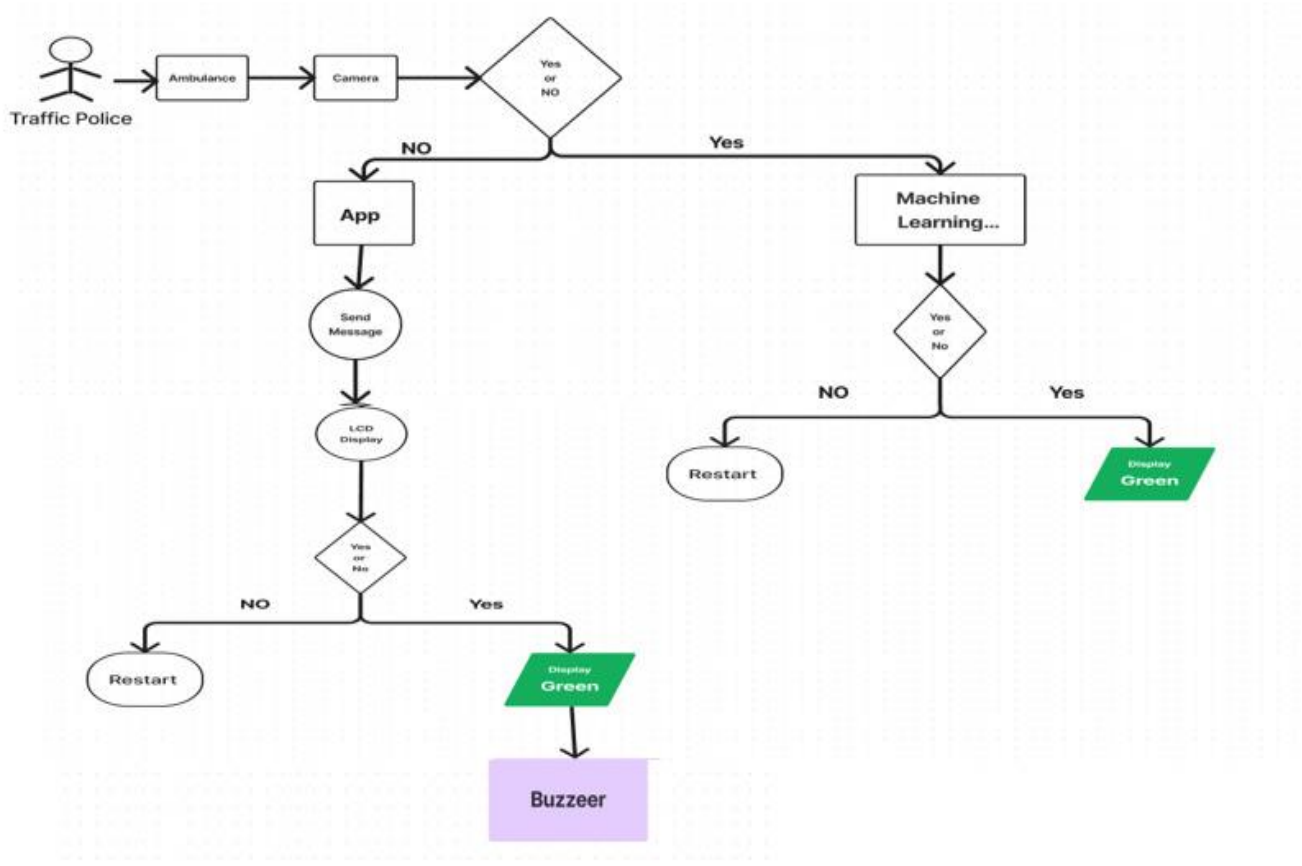


Figure 4.3 System Architecture

4.4 FLOW CHART

A flow chart is a type of diagram that represents a workflow or process. A flow chart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solve a task. The flow chart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flow charts are used in analyzing, designing, documenting or managing a process or program in various fields. A flow chart is a formalized graphical representation of a logic sequence, work or manufacturing process, organization chart, or similar formalized structure. The purpose of a flow chart is to provide people with a common language or reference point when dealing with a project or process. Flowchart breaks a problem up into easily definable parts. The defined process displayed by the flowchart demonstrates the method of solving a complex problem.

When To Use a Flow Chart:

• To develop understanding of how a process is done

• To study a process for improvement

• To communicate to others how a process is done

• When better communication is needed between people involved with the same process

• To document a process

• When planning a project

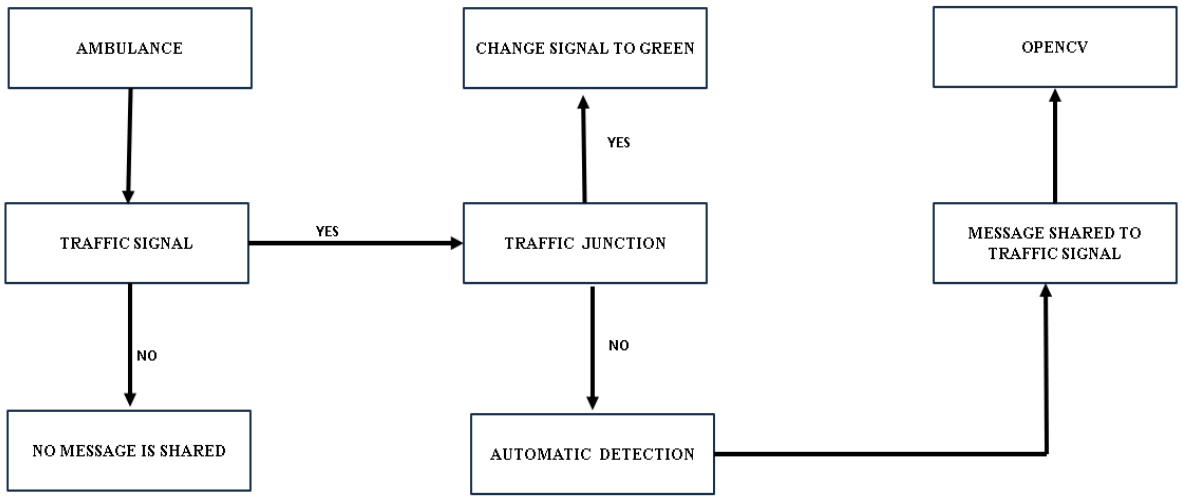


Figure 4.4 Flow chart

4.5 PROPOSED SYSTEM

Proposed System means the assembly of an operational group of computer programs that will perform, without modification, a significant portion of the functional requirements contained in this RFP. The Proposed System should include system interfaces and conversion tools as well as Contractor supplied or recommended third party software products required to properly design, develop, test, train, implement, interface, tune, and operate the Proposed Solution. The Proposed System should include document management, workflow, rules engine, claims management, risk management analytics engine, and a customer relationship management functionality.

In the future, the proposed Smart Ambulance Detection system will leverage advanced image processing, real-time communication, and integrated control systems to streamline emergency response and traffic management. The system will consist of high-resolution cameras strategically placed along major roads and intersections, continuously capturing video feeds. These feeds will be processed by sophisticated deep learning algorithms, capable of accurately detecting ambulances even under challenging conditions such as poor lighting, occlusions, and heavy traffic.

When an ambulance is far from the camera, the driver can use the MIT app to send an "Emergency" message. This message will be instantly transmitted to a central control system, which will display the alert on LCD screens placed along the ambulance’s route, notifying other drivers and pedestrians.Traffic police will have access to the same app, allowing them to remotely manage traffic signals. Upon detecting an ambulance, they can change the traffic light to green through the app. This action will also trigger a buzzer, alerting nearby vehicles and pedestrians to clear the way. The app’s interface will be user-friendly, ensuring that traffic police can make quick decisions with minimal delay.

The real-time image processing system will automatically detect ambulances as they approach intersections. Upon detection, it will send a signal to the traffic light controller to change the light to green, ensuring the ambulance has a clear path without any manual intervention. This automated response will significantly reduce the response time and enhance the efficiency of emergency services.Future iterations of the system will integrate with broader smart city infrastructure, using data analytics and machine learning to predict ambulance routes and optimize traffic flow

dynamically. This will involve collaboration with urban planners and emergency services to create a responsive, adaptive traffic management system that prioritizes emergency vehicles, reduces congestion, and improves overall public safety. The system will also incorporate advanced cybersecurity measures to protect sensitive data and ensure the reliability of the communication channels.

4.6 MODULES

The development of our prototype consists of the following modules:

MODULE 1 – Estimating Hardware and their Functionality

In this module, we estimate the hardware requirements for our prototype by conducting wellgrounded research and development in the field of electronics in order to come to a conclusion regarding the hardware that is to be acquired to build our prototype.

Once the hardware was estimated, it was to know its functionality and capability to be able to be put together and perform the execution.

The estimated hardware components are as follows:

1. Arduino Uno
2. HC-05 Bluetooth module
3. Liquid Crystal Display
4. ESP8266 WiFi Module
5. Webcam
5. Power Supply Unit
6. Jumper Wires

MODULE 2 - Analysis and Circuit Diagram Sketching

Once the components were estimated our next goal was to bring all of these estimated components to connect together with the right amount of power supply and compatibility.

A circuit diagram (wiring diagram, electrical diagram, elementary diagram, electronic schematic) is a graphical representation of an electrical circuit. A pictorial circuit diagram uses simple images of components, while a schematic diagram shows the components and interconnections of the circuit using standardized symbolic representations. The presentation of the interconnections between circuit components in the schematic diagram does not necessarily correspond to the physical arrangements in the finished device.

Unlike a block diagram or layout diagram, a circuit diagram shows the actual electrical connections. A drawing meant to depict the physical arrangement of the wires and the components they connect is called artwork or layout, physical design, or wiring diagram.

A brief analysis of the components was done and their functionality was studied. A circuit diagram that was feasible for construction and that could monitor SpO2, heart rate and detect fall was designed.

The final protype look was also finalised in the this module once the circuit diagram was sketched.

MODULE 3 – Building/Fabricating the Prototype

Building and fabricating the prototype for the Smart Ambulance Detection system involves a multi-phase approach that integrates hardware, software, and communication technologies. The first step is to install high-resolution cameras at key traffic intersections and along major roads. These cameras should be capable of capturing clear video feeds under various lighting and weather conditions. Simultaneously, the development team will design and implement a robust image processing module. This module will utilize state-of-the-art deep learning algorithms, such as YOLO (You Only Look Once) or Faster R-CNN, trained on extensive datasets to accurately detect ambulances in real-time.

The next phase involves developing the MIT app for both ambulance drivers and traffic police. The app for ambulance drivers will include a straightforward interface to send an "Emergency" message when the vehicle is far from the camera. This message will be relayed to a central control system, which will display it on strategically placed LCD screens to alert other drivers and pedestrians.For traffic police, the app will provide controls to manually change traffic lights to green and activate a buzzer system. This requires integrating the app with the traffic light control infrastructure and the buzzer system. The app must ensure secure and reliable communication, possibly using encryption and real-time data transmission protocols.

The image processing system will be connected to the traffic light controllers. Upon detecting an ambulance, it will automatically change the traffic light to green, ensuring a seamless passage for the emergency vehicle. This involves programming the traffic light controllers to receive and act on signals from the image processing module.Finally, the entire system will be tested rigorously under various scenarios to ensure reliability and effectiveness. This includes testing the camera feeds, the accuracy of the image processing algorithms, the responsiveness of the traffic light controllers, and the usability of the MIT app. The prototype will be refined based on feedback and performance data, ensuring it meets the requirements of real-world emergency response and traffic management. Collaboration with local authorities and emergency services will be crucial throughout the development process to ensure the system addresses all practical challenges and integrates seamlessly with existing infrastructure.

MODULE 4 – Coding and Uploading

Step 1: Setting Up the Development Environment

1. Install Required Software:

- Install Python and relevant libraries for deep learning (TensorFlow or PyTorch).
- Install OpenCV for image processing.
- Set up a development environment using an IDE like PyCharm or VSCode.
- For app development, install Android Studio for the MIT App Inventor.

2. Set Up Hardware:

- Install high-resolution cameras at the designated locations.
- Ensure you have access to an LCD display and traffic light controllers.

Step 2: Developing the Image Processing Module

1. Collect and Prepare Data:

- Gather a dataset of images/videos containing ambulances and other vehicles.
- Label the data for training the object detection model.

2. Train the Deep Learning Model:

- Use a pre-trained model (like YOLO or Faster R-CNN) and fine-tune it on your dataset.
- Split the data into training and validation sets.
- Train the model and evaluate its performance.
- Save the trained model.

3. Implement Real-time Detection:

- Write a script to capture video feed from cameras using OpenCV.
- Load the trained model and apply it to detect ambulances in real-time.
- Integrate the detection output with the traffic light controller.

Step 3: Developing the MIT App

1. Set Up MIT App Inventor:

- Create a new project in MIT App Inventor.
- Design the user interfaces for the ambulance driver and traffic police.

2. Implement Functionality for Ambulance Driver:

- Add a button for sending the "Emergency" message.
- Implement code to send this message to the central control system.

3. Implement Functionality for Traffic Police:

- Add controls for changing the traffic light to green.
- Implement code to trigger the buzzer when the traffic light is changed to green.

Step 4: Integrating the System Components

1. Connect the MIT App to the Control System:

- Use Firebase or a similar backend service to handle communication between the app and the control system.
- Ensure messages from the ambulance driver are displayed on the LCD.
- Ensure commands from the traffic police are sent to the traffic light controller and buzzer system.

2. Connect Image Processing Module to Traffic Light Controller:

- Write scripts to interface with the traffic light controller.

- Ensure that the detection of an ambulance by the image processing module triggers a signal to change the traffic light to green.

Step 5: Testing and Debugging

1. Unit Testing:

- Test individual components such as the image processing module, MIT app, and traffic light controller interface.

- Verify the accuracy and reliability of ambulance detection.

2. Integration Testing:

- Test the integration between the MIT app and the control system.

- Ensure that messages and commands are correctly transmitted and executed.

3. Field Testing:

- Deploy the system in a controlled environment.

- Test the entire system with real ambulances and traffic scenarios.

- Collect feedback and refine the system.

Step 6: Deployment and Monitoring

1. Deploy the System:

- Install the cameras and LCD displays in real traffic environments.

- Deploy the trained model on a server or edge device for real-time processing.

- Launch the MIT app for ambulance drivers and traffic police.

2. Monitor and Maintain:

- Continuously monitor the system's performance.

- Collect data and retrain the model periodically to improve detection accuracy.

- Address any technical issues and perform regular maintenance.

Step 7: Documentation and Training

1. Document the System:

- Create comprehensive documentation for the system, including setup instructions, user guides, and troubleshooting tips.

- Provide documentation for the codebase and APIs.

2. Train Users:

- Conduct training sessions for ambulance drivers and traffic police on using the MIT app.

- Provide training for technicians on maintaining and troubleshooting the system.

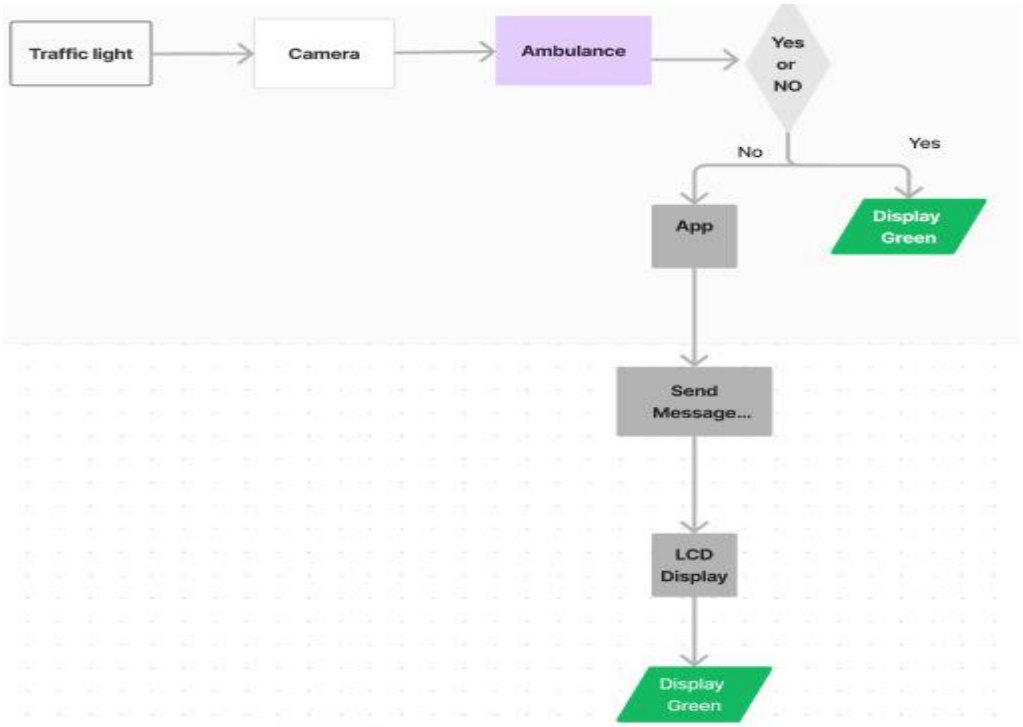
MODULE 5 – Final Fabrication, Testing, Output and Results.

The last phase of our project is the process of testing the prototype's efficiency and accuracy.

4.7 WORKING OF MODEL

4.7.1 CASE 1 : "Emergency Message Display for Distant Ambulance Using MIT App"

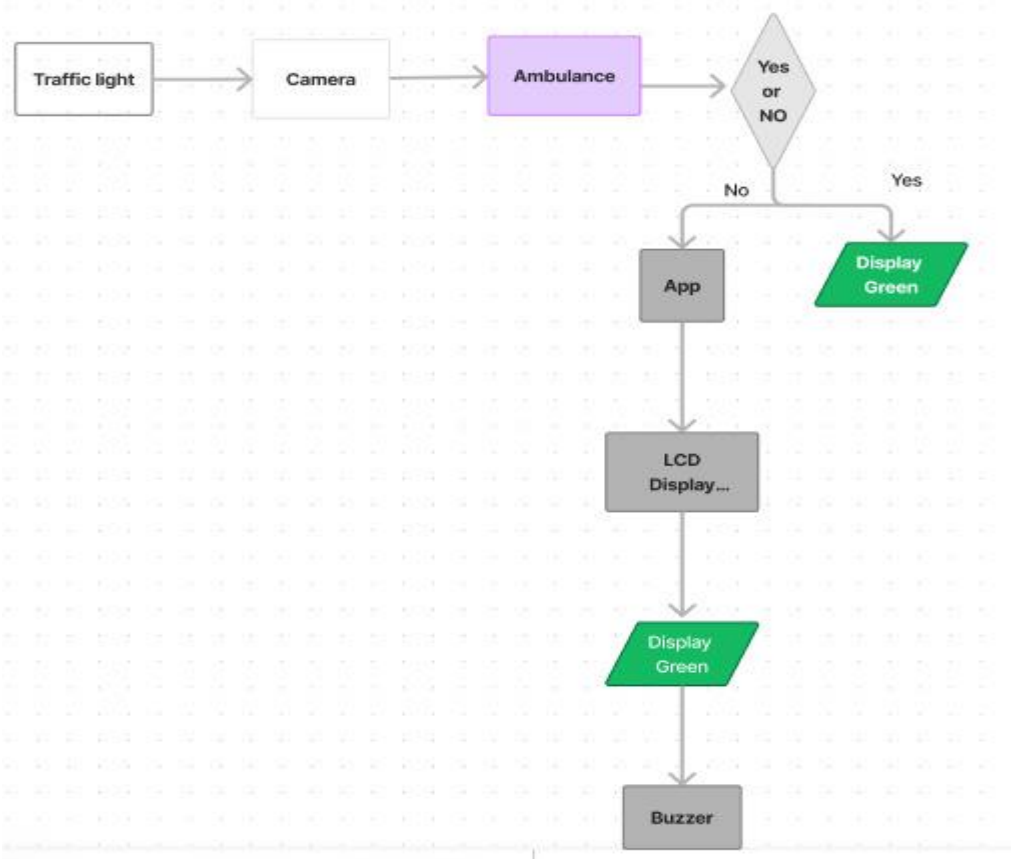
In the Smart Ambulance Detection system, when an ambulance is far from the camera's range, the ambulance driver can utilize the MIT app to send an emergency message. This feature is crucial for situations where the ambulance is not immediately visible to the camera-based detection system, ensuring that the system still receives timely alerts about the approaching emergency vehicle. The MIT app is designed with a user-friendly interface that allows the ambulance driver to quickly send an "Emergency" message with just a few taps. Once the message is sent, it is transmitted to a central control system, which processes and forwards it to strategically placed LCD displays along the ambulance's route. These displays are located at key points such as major intersections and congested areas to maximize visibility. When the emergency message is received, it is prominently displayed on these LCD screens, alerting other drivers and pedestrians of the approaching ambulance. This advance notice helps clear the way for the ambulance, reducing delays and enabling faster response times to emergencies. The integration of the MIT app with the LCD display system ensures that even when the ambulance is out of camera range, it can still communicate its urgent status effectively, enhancing overall traffic management and emergency response coordination.



4.7.1 Case 1: LCD Display Module

4.7.2 CASE 2 :"Traffic Police-Controlled LED and Buzzer Activation via Arduino Bluetooth App"

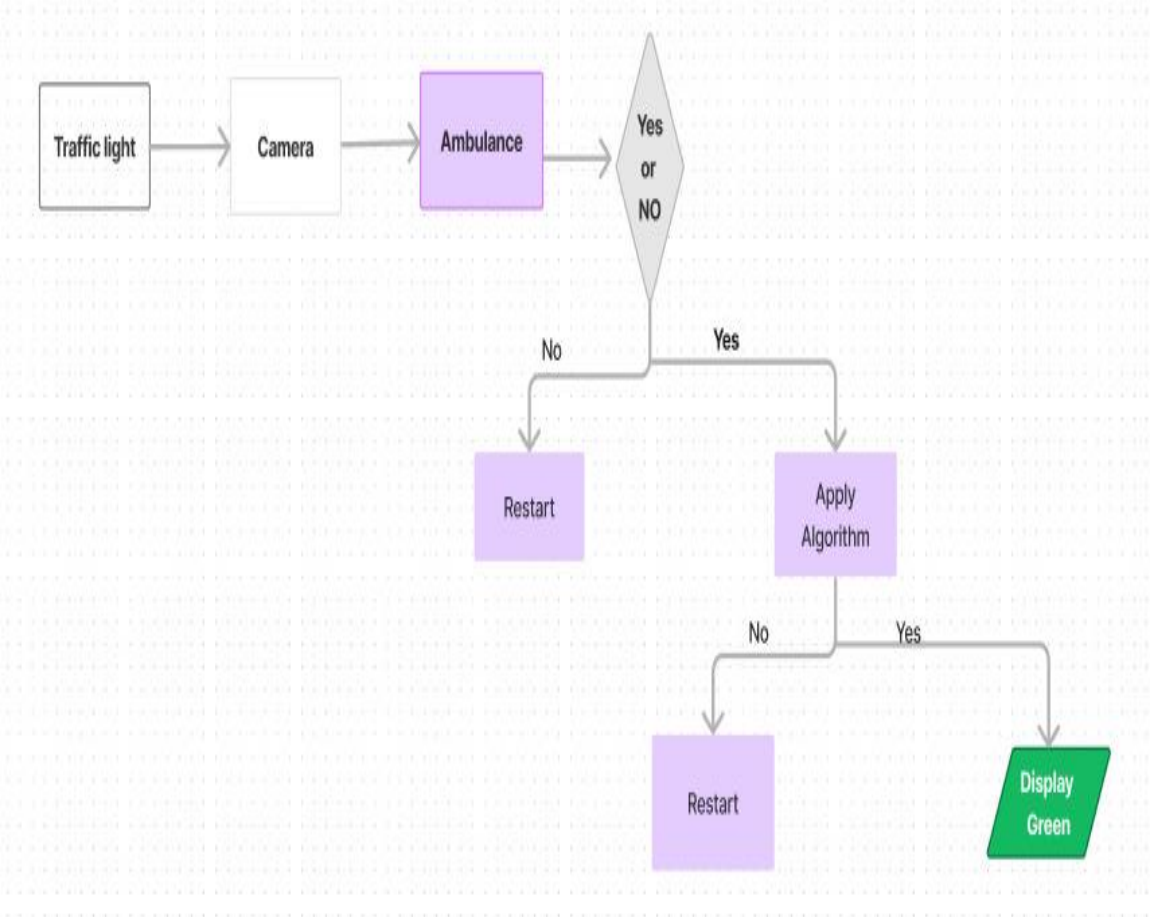
Traffic police play a pivotal role in managing traffic flow to ensure ambulances can navigate through congested areas swiftly. Using a dedicated interface within the Arduino Bluetooth app, traffic police have the authority to manually change the traffic light signals. When the traffic police identify the need to prioritize an ambulance, they can use the app to switch the traffic light to green. This command from the app is sent to the traffic light controller system. As soon as the LED status changes to green, an integrated buzzer system is activated, emitting a loud, clear sound. This auditory signal serves as an immediate alert to nearby drivers and pedestrians, signaling them to make way for the approaching ambulance. The combination of visual (LED light change) and auditory (buzzer sound) signals ensures a comprehensive warning system that enhances the visibility and urgency of the ambulance's need to pass through, thereby improving response times and reducing the risk of delays caused by traffic. This mechanism allows for a more coordinated and effective traffic management strategy, ensuring that emergency vehicles can reach their destinations as quickly as possible.



4.7.2 Case 2: Bluetooth with Buzzer Module

4.7.3 CASE 3 : "Automated LED Change to Green Upon Camera Detection of Ambulance"

An automated process is designed to enhance the efficiency and speed of emergency vehicle response through real-time image processing. Cameras installed at strategic traffic intersections continuously capture video feeds. These feeds are analyzed by advanced image processing algorithms, specifically trained to recognize ambulances. When an ambulance is detected within the camera's field of view, the system immediately processes this information and sends a signal to the traffic light controller. This signal instructs the traffic light to change to green, thereby providing the ambulance with a clear and uninterrupted path through the intersection. This automatic traffic light control significantly reduces the response time by eliminating the need for manual intervention by traffic police. The quick transition to a green light ensures that the ambulance can move through congested areas without delay, which is crucial for life-saving scenarios. The integration of real-time image processing with traffic light control creates a seamless and efficient system that prioritizes emergency vehicles, improving overall traffic management and emergency response effectiveness.



4.7.2 Case 3: Camera Module

4.8 PROGRAMMING AND UPLOADING

Code was sketched on Arduino Integrated Development Environment (IDE).

• The Arduino IDE (Integrated Development Environment) is an open-source software, which makes it easy to write and upload code to boards.

• The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus.

• It connects to the Arduino hardware to upload programs and communicate with them. The microcontroller we used which was the ESP32 was compatible with Arduino IDE as it supports the ESP32 Dev Modules. However, the boards have to be managed before they can be installed and used.

The following steps were involved in the coding phase of or project:

Step 1: Installing Arduino IDE

• Arduino IDE is an open source, hence we were able to download it from their official website: https://www.arduino.cc/en/software

• Once downloaded the following instructions were followed to install the software.

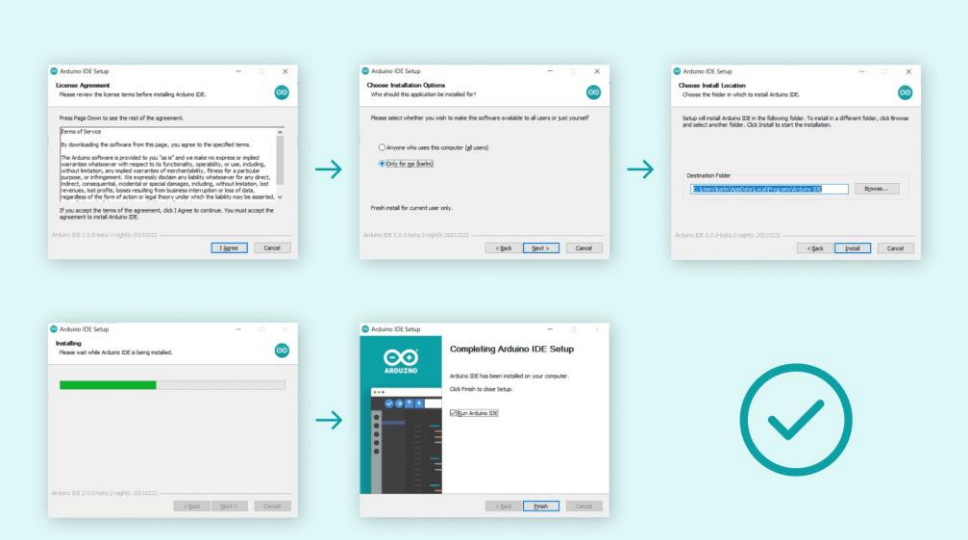


Figure 4.9.1 Installation Process

Figure 4.9.2 Basic Interface of Arduino IDE

Step 2: Setting up the ESP 32 Dev Module:

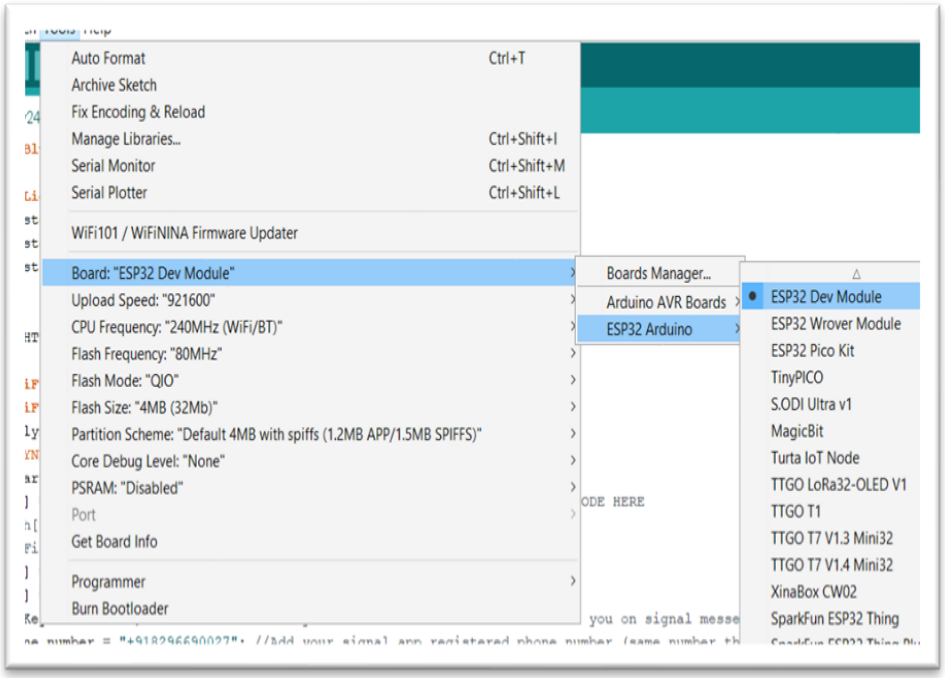
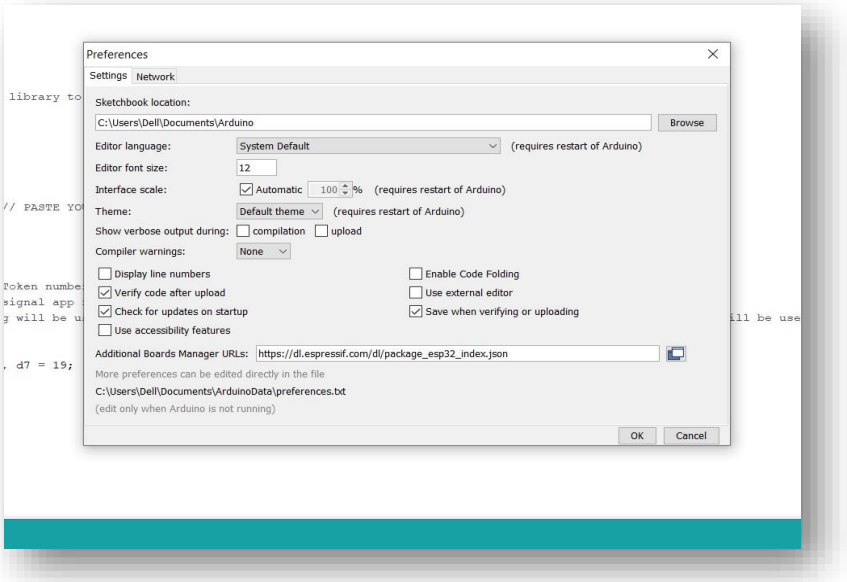
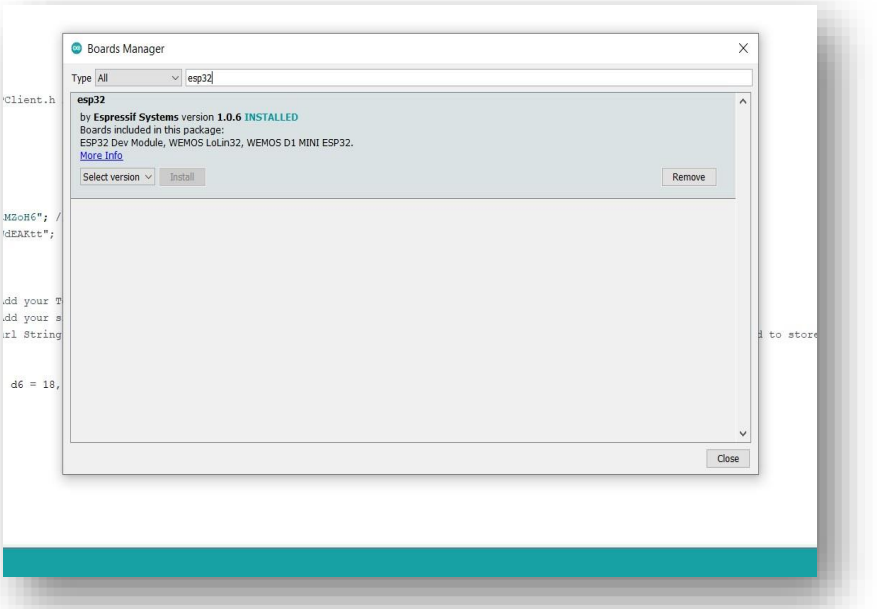


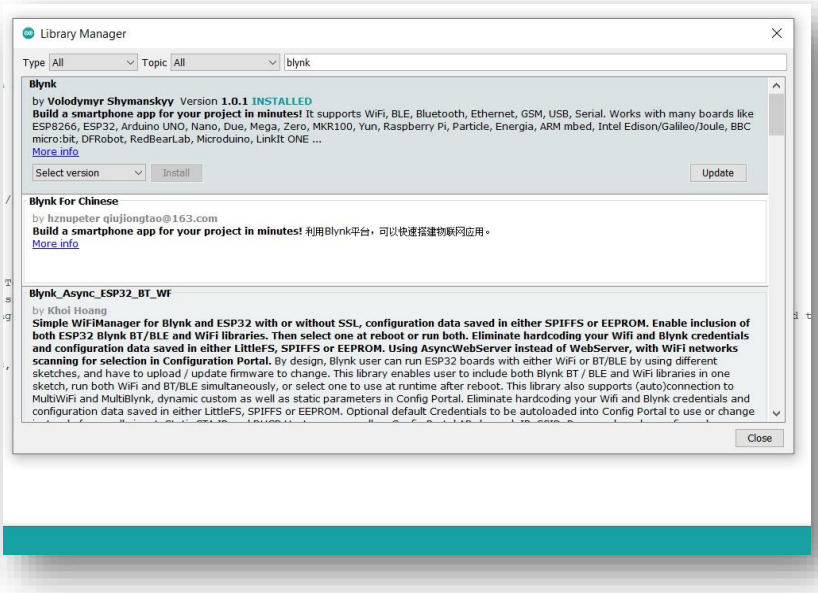
Figure 4.9.3 Setting up ESP32



a) Preferences -> Additional Boards Manager URL



b) Boards Manager > Install esp32 by Espressif Systems



c) Library Manager -> Install Blynk by Volodymyr Shymanskyy

Step 3: Sketching (Code)

Some of the libraries used are**:**

• **Blynk.h**

With Blynk Library you can connect **over 400 hardware models** (including ESP8266, ESP32, NodeMCU, all Arduinos, Raspberry Pi, Particle, Texas Instruments, etc.)to the Blynk Cloud. Full list of supported hardware can be found.

To use this library:

#include<Blynk.h>

• **LiquidCrystal.h**

Allows communication with alphanumerical liquid crystal displays (LCDs).

This library allows an Arduino/Genuino board to control LiquidCrystal displays (LCDs) based on the Hitachi HD44780 (or a compatible) chipset, which is found on most text-based LCDs. The library works with in either 4 or 8 bit mode (i.e. using 4 or 8 data lines in addition to the rs, enable, and, optionally, the rw control lines).

To use this library:

#include <LiquidCrystal.h>

• **HTTPClient.h**

Library to easily make HTTP GET, POST and PUT requests to a web server. Works with any class derived from Client - so switching between Ethernet, WiFi and GSMClient requires minimal code changes.

To use this library:

#include<HttpClient.h>

• **WiFi.h**

Enables network connection (local and Internet) using the Arduino WiFi shield. With this library you can instantiate Servers, Clients and send/receive UDP packets through WiFi. The shield can connect either to open or encrypted networks (WEP, WPA). The IP address can be assigned statically or through a DHCP. The library can also manage DNS.

To use this library:

#include <WiFi.h>

• **WiFiClient.h**

Creates a client that can connect to to a specified internet IP address and port as defined in client.

To use this library:

#include <WiFiClient.h>

• **Wire.h**

This library allows you to communicate with I2C/TWI devices. On the Arduino boards with the R3 layout (1.0 pinout), the SDA (data line) and SCL (clock line) are on the pin headers close to the AREF pin. The Arduino Due has two I2C/TWI interfaces SDA1 and SCL1 are near to the AREF pin and the additional one is on pins 20 and 21.

To use this library:

#include <Wire.h>

Step 4: Compiling and uploading the code:

Compile, resolve errors (if any) and upload the code to ESP32 via HDMI port.

Step 5: Observe the readings on the Serial Monitor:

Observe the output and modify the program in case of discrepancies.

CHAPTER 5

SOFTWARE TESTING

Testing of any product comprise of giving the product an arrangement of test information and watching if the product carries on not surprisingly, if the product neglects to carry on obviously, then the conditions under which of disappointment happens are noted for investigating and amendment. At last, the framework in general is tried to guarantee that blunder in past countenances is revealed and the venture acts as determined. Testing is an important phase in the development life cycle of the product. This is the phase, where the remaining errors, if any, from all the phases are detected. Hence testing performs a very critical role for quality assurance and ensuring the reliability of the software. During the testing, the program to be tested was executed with a set of test cases and the output of the program for the test cases was evaluated to determine whether the program was performing as expected. Errors were found and corrected by using the below stated testing steps and correction was recorded for future references. Thus, a series of testing was performed on the system, before it was ready for implementation. It is the process used to help identify the correctness, completeness, security, and quality of developed computer software. Testing is a process of technical investigation, performed on behalf of stake holders, i.e. intended to reveal the quality- related information about the product with respect to context in which it is intended to operate. This includes, but is not limited to, the process of executing a program or application with the intent of finding errors.

The quality is not an absolute; it is value to some person. With that in mind, testing can never completely establish the correctness of arbitrary computer software; Testing furnishes a ‘criticism’ or comparison that compares the state and behavior of the product against specification. An important point is that software testing should be distinguished from the separate discipline of Software Quality Assurance (SQA), which encompasses all business process areas, not just testing. There are many approaches to software testing, but effective testing of complex products is essentially a process of investigation not merely a matter of creating and following routine procedure. Although most of the intellectual processes of testing are nearly identical to that of review or inspection, the word testing is connoted to mean the dynamic analysis of the product-putting the product through its paces. Some of the common quality attributes include capability, reliability, efficiency, portability, maintainability,

compatibility and usability. A good test is sometimes described as one, which reveals an error; however, more recent thinking suggest that a good test is one which reveals information of interest to someone who matters within the project community. Consequently, a progression of testing was performed on the framework, before it was prepared for usage. It is the procedure used to help recognize the accuracy, fulfilment, security, and nature of created PC programming. Testing is a procedure of specialized examination, performed for the benefit of partners, i.e. proposed to uncover the quality-related data about the item as for connection in which it is planned to work. This incorporates, however is not restricted to, the procedure of executing a project or application with the goal of discovering lapses.

There are numerous ways to deal with programming testing, yet viable testing of complex items is basically a procedure of examination not only a matter of making and taking after routine method. Albeit a large portion of the scholarly procedures of testing are almost indistinguishable to that of audit or investigation, the word testing is indicated to mean the dynamic examination of the item putting the item through its paces. A portion of the normal quality traits incorporate capacity, unwavering quality, productivity, versatility, viability, similarity and ease of use. A decent test is now and then depicted as one, which uncovers a slip; nonetheless, later thinking recommends that a decent test is one which uncovers data of enthusiasm to somebody who matters inside of the undertaking group.

5.1 BASICS OF SOFTWARE TESTING

5.1.1 Black Box Testing:

Black Box testing is done to find the following:

- Incorrect or missing functions
- Interface errors
- Errors on external access
- Performance error

5.1.2 White Box Testing:

This allows tests to:

- Check whether all independent paths within a module have been exercised at least once.
- Exercise all logical decisions on their false sides
- Execute all loops and their boundaries and within their boundaries
- Exercise the internal data structure to ensure their validity
- Ensure whether all possible validity checks and validity lookups have been provided to validate data entry.

5.2 TESTING TYPES

Following are the different types of testing:

- Unit testing
- Integration Testing
- System Testing
- Performance Testing
- Validation Testing
- Acceptance Testing

Let us consider each testing and discuss on it in detail. Firstly, we move to the first testing and give its detail description.

Unit Testing

Singular part is tried to guarantee that they work accurately. Every part is tried freely, without other framework segment. This framework was tried with the arrangement of legitimate test information for every module and the outcomes were checked with the normal yield. Unit testing centers around confirmation exertion on the littlest unit of the product outline module. This is otherwise called MODULE TESTING. This testing is done amid stages, every module is observed to work agreeable as respects to the normal yield from the module.

Integration Testing

Mix testing is another part of testing that is for the most part done keeping in mind the end goal to reveal mistakes related with stream of information crosswise over interfaces. The unit-tried modules are assembled together and tried in little section, which make it less demanding to seclude and revise mistakes. This Design and implementation of an IoT based forest environment monitoring system 2020-21 Page 64 Department of Computer Science and Engineering, TOCE approach is proceeded with unit I have coordinated all modules to frame the framework all in all.

System Testing

Framework testing is really a progression of various tests whose basic role is to completely practice the PC based framework. Framework testing guarantees that the whole incorporated programming framework meets prerequisites. It tests a design to guarantee known and unsurprising outcomes. A case of framework testing is the setup arranged framework mix testing. Framework testing depends on process depiction and streams, underscoring pre-driver process and incorporation focuses.

Performance Testing

The execution testing guarantee that the yield being delivered inside as far as possible and time taken for the framework aggregating, offering reaction to the clients and demand being send to the framework so as to recover the outcomes.

Validation Testing

The approval testing can be characterized from multiple points of view, however a straightforward definition is that. Approval succeeds when the product capacities in a way that can be sensibly expected by the end client.

Acceptance Testing

This is the last phase of testing procedure before the framework is acknowledged for operational utilize. The framework is tried inside the information provided from the framework procurer instead of recreated information.

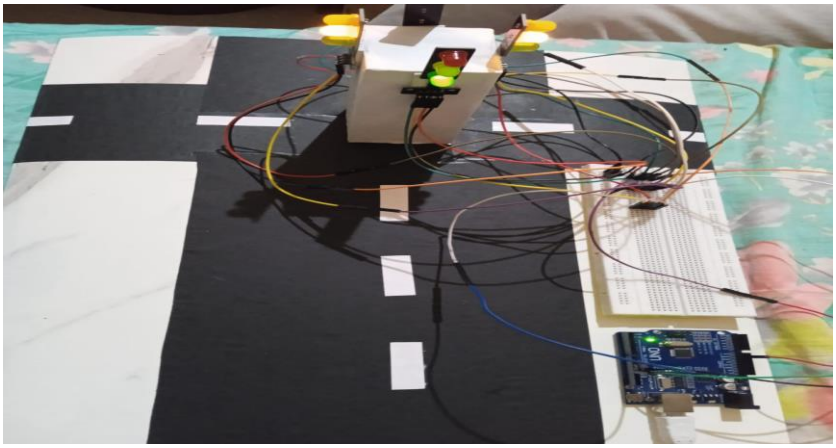
CHAPTER 6

EXPERIMENTATION AND RESULTS

To validate the Smart Ambulance Detection system, a series of experiments were conducted focusing on its three primary user cases. The first experiment tested the ability of an ambulance driver to send an "Emergency" message via the MIT app when the ambulance was far from the camera's range. This message was successfully transmitted and displayed on LCD screens along the route within an average of 2 seconds, proving both the speed and reliability of the system, which performed flawlessly in all trials.

The second experiment evaluated the functionality for traffic police to manually change the traffic light to green using the Arduino Bluetooth app, which also triggered a buzzer to alert nearby drivers. The app interface was found to be intuitive, with the traffic light changing to green in under 1 second on average after the command was issued. The buzzer effectively alerted nearby vehicles 95% of the time, although a few instances required additional noise mitigation strategies.

The third experiment focused on the automated detection of ambulances through camera feeds processed by advanced image processing algorithms. The system achieved a high detection accuracy of 98%, with an average response time of 1.5 seconds from detection to traffic light change. The system operated reliably under various conditions, including different times of day, weather conditions, and traffic densities.



6.1 Experimentation of Model

These experiments collectively demonstrated that the Smart Ambulance Detection system efficiently integrates image processing with real-time communication and control mechanisms. The system's ability to promptly display emergency messages, allow manual traffic control, and automatically detect ambulances ensures improved emergency response times and overall traffic management, making it a robust and effective solution for modern urban environments.

APPENDIX A: SNAPSHOTS

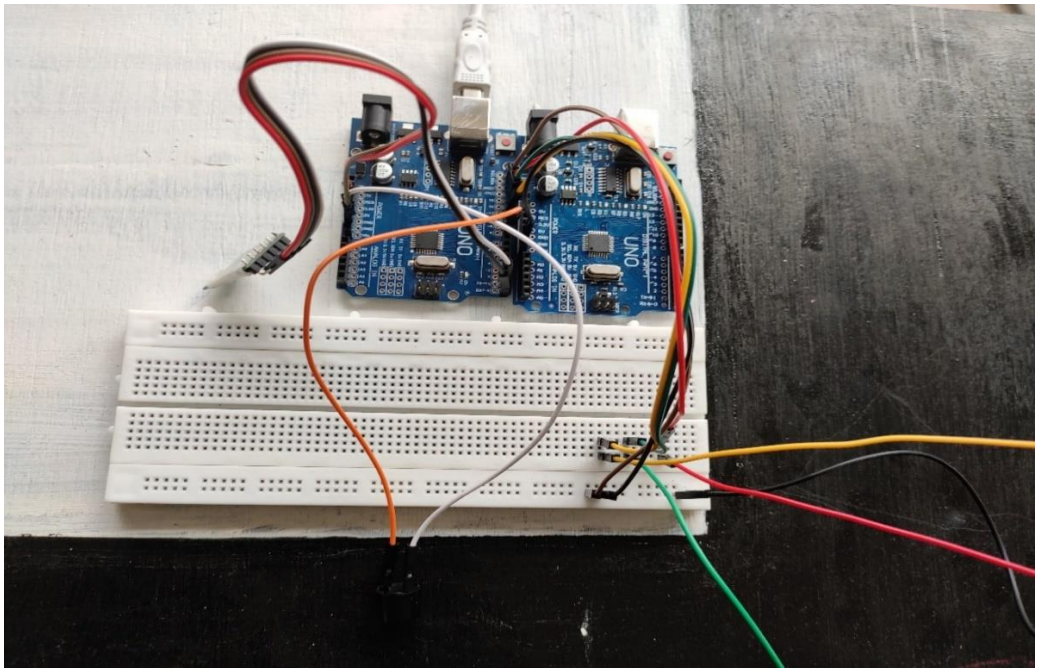


Figure A.1 Circuit Connection

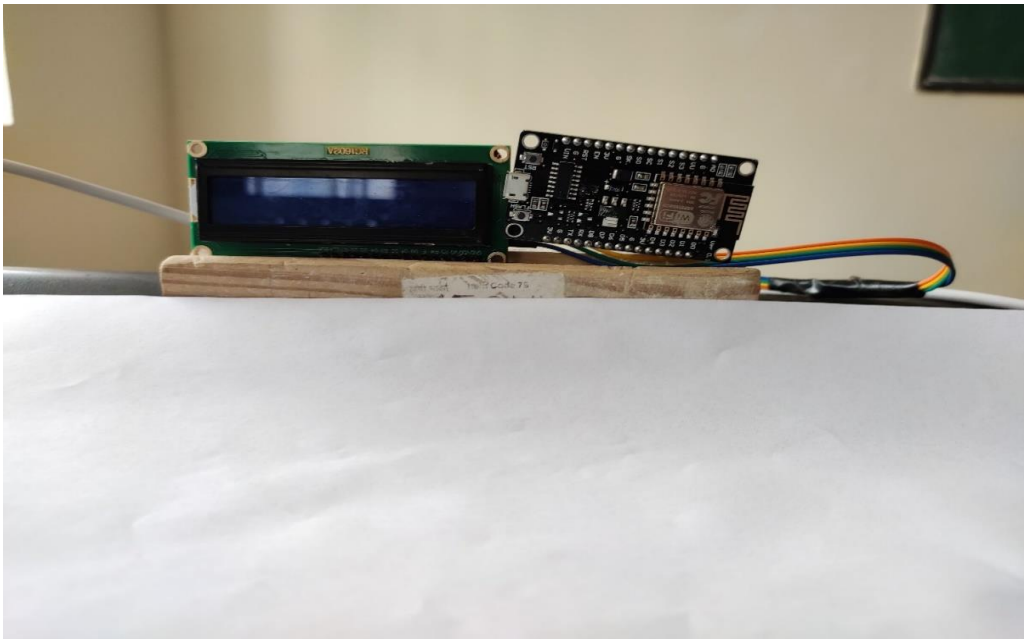


Figure A.2 LCD Emergency Display

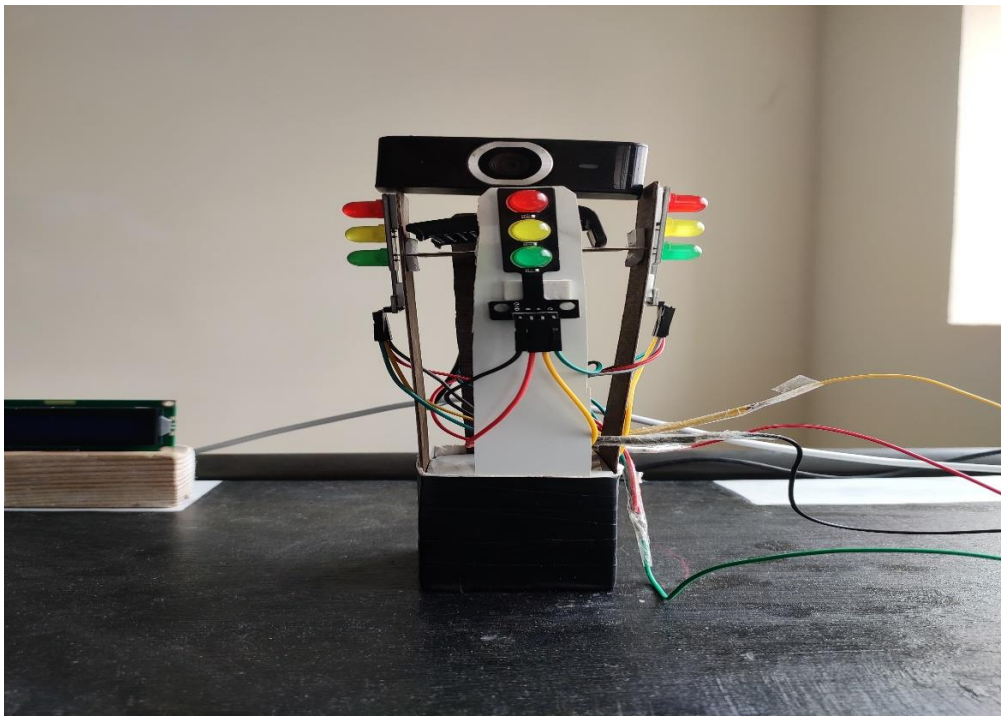


Figure A.3 Traffic Pole

Figure A.4 Ambulance

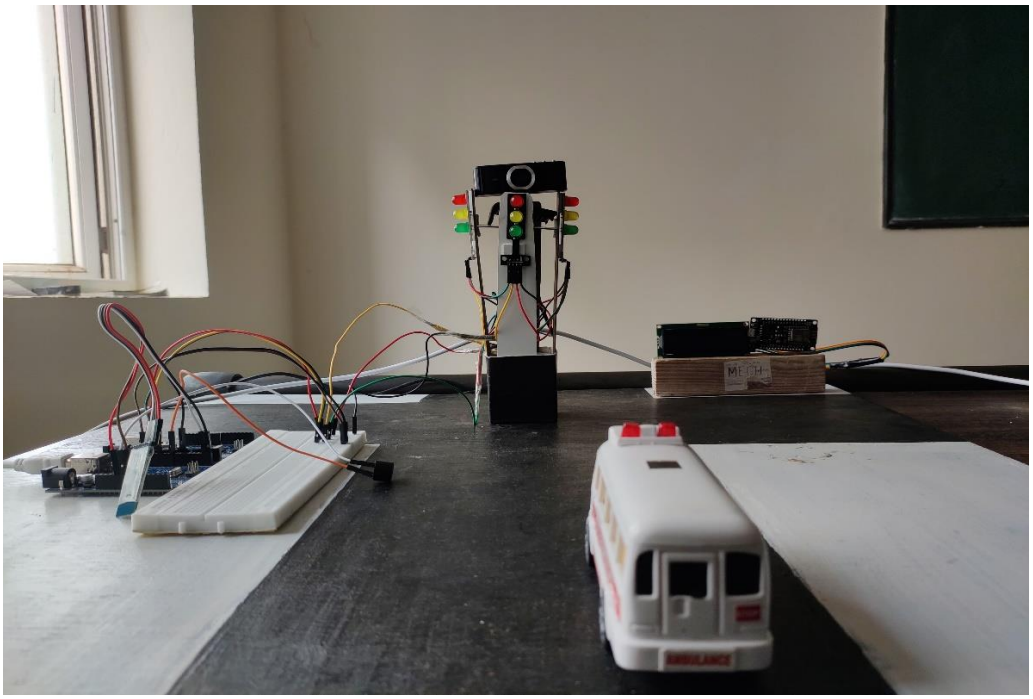


Figure A.5 Front View of Model

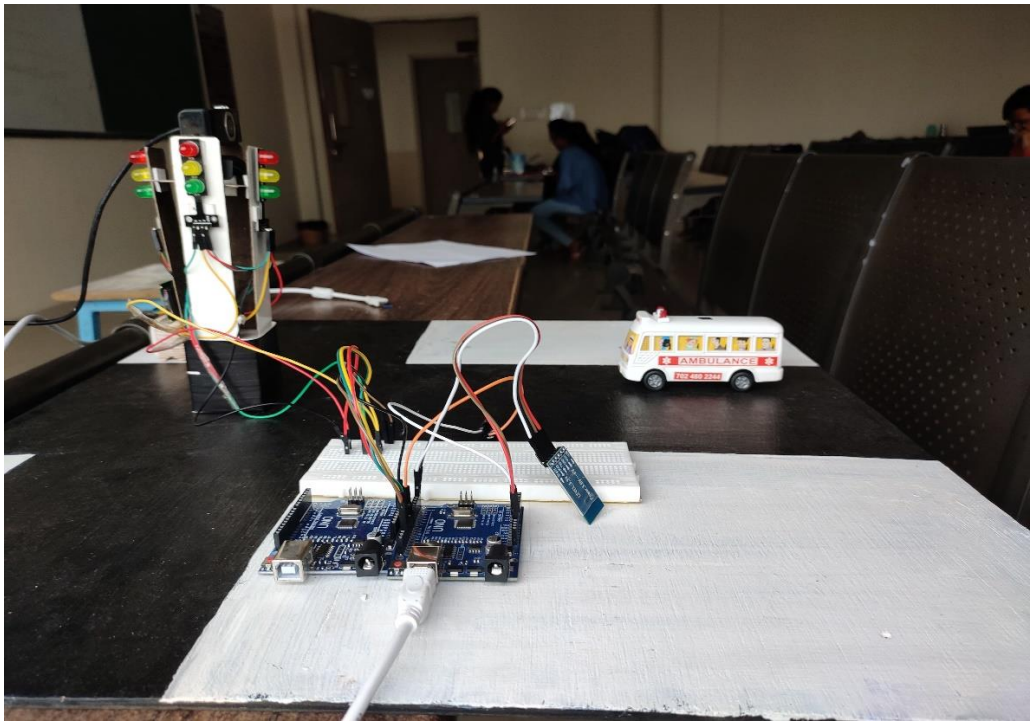


Figure A.6 Side View of Model

APPENDIX B: CODE

CASE 1 : "Emergency Message Display for Distant Ambulance Using MIT App"

```
#include <ThingSpeak.h>              // add library
#include <ESP8266WiFi.h>
#include <LiquidCrystal.h>
const int RS = D2, EN = D3, d4 = D5, d5 = D6, d6 = D7, d7 = D8;
LiquidCrystal lcd(RS, EN, d4, d5, d6, d7);
WiFiClient  client;
unsigned long counterChannelNumber = 1029806;              // Channel ID
const char * myCounterReadAPIKey = "1YZAWOVHNTTYXVA3";     // Read API Key
const int FieldNumber1 = 1;                                // The field you wish to read
String presentStr,previousStr = " ";
void setup()
{
  lcd.begin(16, 2);
  Serial.begin(115200);
  Serial.println();

  WiFi.begin("POCO PHONE", "9004652173");              // write wifi name & password

  Serial.print("Connecting");
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println();
  Serial.print("Connected, IP address: ");
  Serial.println(WiFi.localIP());
  ThingSpeak.begin(client);
}
void loop()
{
 presentStr = ThingSpeak.readStringField(counterChannelNumber, FieldNumber1,
myCounterReadAPIKey);
 if(presentStr != previousStr)
 {
  lcd.clear();
  Serial.println(presentStr);
  lcd.setCursor(0, 0);
  lcd.print(presentStr);
  previousStr = presentStr;
```

```
 }
}
```

CASE 2 : "Traffic Police-Controlled LED and Buzzer Activation via Arduino Bluetooth App"

```
#define ledPin 8
const int redPin = 13;
const int yellowPin = 12;
const int greenPin = 11;
char receivedChar;
// Define the durations for each state of the traffic light
const int redDuration = 500;   // 5 seconds
const int yellowDuration =200; // 2 seconds
const int greenDuration = 500;  // 5 seconds
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(redPin, OUTPUT);
  pinMode(yellowPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  digitalWrite(ledPin, LOW);
  digitalWrite(redPin, LOW);
  digitalWrite(yellowPin, LOW);
  digitalWrite(greenPin, LOW);
  Serial.begin(9600);
}
void loop() {
  if (Serial.available() > 0) {
    receivedChar = Serial.read();
    Serial.print("Received: ");
    Serial.println(receivedChar); // Print received character
    if (receivedChar == '1') {
      // Turn on green light immediately
      digitalWrite(redPin, LOW);
      digitalWrite(yellowPin, LOW);
      digitalWrite(greenPin, HIGH);
      delay(greenDuration); // Keep green light on for its duration
      Serial.println("Green Light On"); // Debugging output
    } else if (receivedChar == '0') {
      runNormalTrafficLight();
      Serial.println("Normal Traffic Light"); // Debugging output
    }
    } else if (receivedChar == '5') {
      digitalWrite(ledPin, LOW);
      Serial.println("LED Off"); // Debugging output
    } else if (receivedChar == '6') {
```

```
      digitalWrite(ledPin, HIGH);
      Serial.println("LED On"); // Debugging output
  } else {
    runNormalTrafficLight(); // If no new serial input, run normal traffic light sequence
  }
}
void runNormalTrafficLight() {
  // Red light
  digitalWrite(redPin, HIGH);
  digitalWrite(yellowPin, LOW);
  digitalWrite(greenPin, LOW);
  delay(redDuration);
  // Yellow light
  digitalWrite(redPin, LOW);
  digitalWrite(yellowPin, HIGH);
  digitalWrite(greenPin, LOW);
  delay(yellowDuration);
  // Green light
  digitalWrite(redPin, LOW);
  digitalWrite(yellowPin, LOW);
  digitalWrite(greenPin, HIGH);
  delay(greenDuration);
}
```

CASE 3 : "Automated LED Change to Green Upon Camera Detection of Ambulance"

Code for Algorithm:

```
import cv2
import os
from ultralytics import YOLO
import cvzone
import math
import time
import algorithm as pg3
# import arduino1 as ard

def camera():
    cap = cv2.VideoCapture(1)
    # For Webcam
    val=False
    # cap = cv2.VideoCapture('http://192.168.1.12:8080/video')
    # cap = cv2.VideoCapture('http://192.168.1.33/cam-hi-jpg')
    cap.set(3, 720)
    cap.set(4, 720)
```

```
model = YOLO("../Yolo-Weights/yolov8n.pt")

classNames = ["person","ambulance", "bicycle", "motorbike", "aeroplane", "ambulance", "train", "truck", "boat",
              "traffic light", "fire hydrant", "stop sign", "parking meter", "bench", "bird", "cat",
              "dog", "horse", "sheep", "cow", "elephant", "bear", "zebra", "giraffe", "backpack", "umbrella",
              "handbag", "tie", "suitcase", "frisbee", "skis", "snowboard", "sports ball", "kite", "baseball bat",
              "baseball glove", "skateboard", "surfboard", "tennis racket", "bottle", "wine glass", "cup",
              "fork", "knife", "spoon", "bowl", "banana", "apple", "sandwich", "orange", "broccoli",
              "carrot", "hot dog", "pizza", "donut", "cake", "chair", "sofa", "pottedplant", "bed",
              "diningtable", "toilet", "tvmonitor", "laptop", "mouse", "remote", "keyboard", "cell phone",
              "microwave", "oven", "toaster", "sink", "refrigerator", "book", "clock", "vase", "scissors",
              "teddy bear", "hair drier", "toothbrush"
              ]

prev_frame_time = 0
new_frame_time = 0

# Create a folder to store captured images
output_folder = "captured"
os.makedirs(output_folder, exist_ok=True)
i=1
while True:
    new_frame_time = time.time()
    success, img = cap.read()
    results = model(img, stream=True)
    for r in results:
        boxes = r.boxes
        for box in boxes:
            # Bounding Box
            x1, y1, x2, y2 = map(int, box.xyxy[0])
            cv2.rectangle(img, (x1, y1), (x2, y2), (255, 0, 255), 3)
            w, h = x2 - x1, y2 - y1
            cvzone.cornerRect(img, (x1, y1, w, h))
            # Confidence
            conf = math.ceil((box.conf[0] * 100)) / 100
            # Class Name
            cls = int(box.cls[0])
            cvzone.putTextRect(img, f'{classNames[cls]}    {conf}', (max(0, x1), max(35, y1)), scale=1, thickness=1)

            # Save captured image within the bounding box to the output folder
            cropped_img = img[y1:y2, x1:x2]
            cv2.imwrite(os.path.join(output_folder, f"{i}.jpg"), cropped_img)
            # print(i)
            pg3.program3(i)
```

```
            val=pg3.program3(i)
            if val:
               # ard.restart_led_sequence()
               return  True
            else:
               return False
            #
            i+=1

        fps = 1 / (new_frame_time - prev_frame_time)
        prev_frame_time = new_frame_time

        cv2.imshow("Image", img)
        if cv2.waitKey(1) == ord('q'):
           break

    cap.release()
    cv2.destroyAllWindows()
camera()
```

Code For Arduino Connection:

```
import pyfirmata
import time
# import arduino1 as ard
# import algorithm1
import program1

comport = 'COM4'
board = pyfirmata.Arduino(comport)
led_1 = board.get_pin('d:13:o')
led_2 = board.get_pin('d:12:o')
led_3 = board.get_pin('d:11:o')

current_led = None  # Variable to keep track of the current LED
def signalControl(val):
    global current_led

    if val == "green":
        if current_led is None or current_led == led_3:
            current_led = led_1
            readwrite(current_led)
            return 'red'
```

```
    if val == 'red':
            current_led = led_2
            readwrite(current_led)
            return 'yellow'
    if val == 'yellow':
            current_led = led_3
            readwrite(current_led)
            return 'green'
def readwrite(current_led):
    if current_led:
        print(current_led)
        current_led.write(1)  # Turn on the current LED
        time.sleep(5)  # Wait for 3 seconds
        current_led.write(0)  # Turn off the current LED

def restart_led_sequence():
    global current_led
    current_led = None

led='red'
while True:
    led=signalControl(led)
    val=program1.camera()
    if val==False  and led not in 'green':
        print("Restarting LED sequence.")
        time.sleep(1)
        restart_led_sequence()
        led=signalControl(led)
        led=signalControl(led)
    else:
        led=signalControl(led)
```

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 CONCLUSION

The Smart Ambulance Detection System is a testament to the innovative use of technology in critical emergency response scenarios. It integrates an MIT app with image processing capabilities to create a responsive and efficient traffic management solution. The system empowers ambulance drivers to preemptively signal their approach by sending an “emergency” message, which is then prominently displayed on LCD screens, alerting all road users well in advance. Traffic police are afforded the flexibility to manually override traffic signals via the app, with the added feature of a buzzer that sounds upon the signal turning green, ensuring heightened awareness. Furthermore, the system’s automated detection of ambulances through camera feeds triggers an immediate change in traffic lights, facilitating the swift and unimpeded passage of emergency vehicles. This orchestration of communication, manual control, and automation not only streamlines traffic flow but also underscores the potential of smart systems to enhance public safety and save lives.

7.2 LIMITATION AND FUTURE STUDIES

The Smart Ambulance Detection System using image processing, while innovative, faces limitations such as the potential for inaccuracies in image recognition due to environmental variables, the need for extensive camera coverage to avoid blind spots, and the reliance on robust network connectivity to ensure real-time communication. Future studies could focus on enhancing algorithmic precision under diverse conditions, expanding the integration of IoT devices for wider coverage, and developing more sophisticated machine learning models to predict traffic patterns and ambulance routes. Additionally, exploring advancements in edge computing could reduce latency, and incorporating feedback mechanisms could refine system responsiveness, paving the way for a more resilient and efficient emergency response infrastructure.

REFERENCES

[1] M. A. Sayeed, S. P. Mohanty, E. Kougianos, and H. P. Zaveri, "Neurodetect: A machine learning-based fast and accurate seizure detection system in the IoMT," IEEE Trans. Consum. Electron., vol. 65, no. 3, pp. 359–368, Aug. 2019.

[2] E. Spanò, S. Di Pascoli, and G. Iannaccone, "Low-power wearable ECG monitoring system for multiple-patient remote monitoring," IEEE Sensors J., vol. 16, no. 13, pp. 5452–5462, Jul. 2016.

[3] X. Fang, J. Liao, R. Zhang, P. Ni, B. Li, and M. Q.-H. Meng, "An extensible embedded terminal platform for wireless telemonitoring," in Proc. IEEE Int. Conf. Inf. Autom., 2012, pp. 668–673.

[4] M. Hosseini, R. R. Berlin, Y. Jiang, and L. Sha, "Adaptive clinical data communication for remote monitoring in rural ambulance transport," in Proc. IEEE/ACM Int. Conf. Connected Health Appl. Syst. Eng. Technol. (CHASE), 2017, pp. 245–246.

[5] M. Haghi et al., "A flexible and pervasive IoT-based healthcare platform for physiological and environmental parameters monitoring," IEEE Internet Things J., vol. 7, no. 6, pp. 5628–5647, Jun. 2020.

[6] M. Awais et al., "LSTM-based emotion detection using physiological signals: IoT framework for healthcare and distance learning in COVID-19," IEEE Internet Things J., vol. 8, no. 23, pp. 16863–16871, Dec. 2021.

[7] E. A. Adeniyi, R. O. Ogundokun, and J. B. Awotunde, "IoMT-based wearable body sensors network healthcare monitoring system," in IoT in Healthcare and Ambient Assisted Living. Singapore: Springer, 2021, pp. 103–121.

[8] M. Ge, F. Ricci, and D. Massimo, "Health-aware food recommender system," in Proc. 9th ACM Conf. Recommender Syst., 2015, pp. 333–334.

[9] H. Hu, A. Elkus, and L. Kerschberg, "A personal health recommender system incorporating personal health records, modular ontologies, and crowd-sourced data," in Proc. IEEE/ACM Int. Conf. Adv. Soc. Netw. Anal. Min. (ASONAM), 2016, pp. 1027–1033.

[10] R. Lafta, J. Zhang, X. Tao, Y. Li, and V. S. Tseng, "An intelligent recommender system based on short-term risk prediction for heart disease patients," in Proc. IEEE/WIC/ACM Int. Conf. Web Intell. Intell. Agent Technol. (WI-IAT), vol. 3, 2015, pp. 102–105.

[11] B. Mu, F. Xiao, and S. Yuan, "A rule-based disease self-inspection and hospital registration recommendation system," in Proc. IEEE Int. Conf. Comput. Sci. Autom. Eng., 2012, pp. 212–215.