

Visvesvaraya Technological University

Jnana Sangama, Belagavi - 590018



A Project Work Phase-2 (18CSP83)

Report on

“SMART TRAFFIC MANAGEMENT SYSTEM FOR AMBULANCE”

Project Report submitted in partial fulfilment of the requirement for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Sunil M	(1KG19CS102)
V Yashaswini Naidu	(1KG19CS108)
Vignesh R	(1KG19CS115)
Vishwas P	(1KG19CS117)

Under the guidance of

Mrs. Amitha S

Asst. Professor

**Department of Computer Science & Engineering
KSSEM, Bengaluru-560109**



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

K. S. School of Engineering and Management

#15, Mallasandra, off. Kanakapura Road, Bengaluru – 560109

2022 - 2023



K. S. School of Engineering and Management

#15, Mallasandra, off. Kanakapura Road, Bengaluru - 560109

Department of Computer Science & Engineering

CERTIFICATE

Certified that the Project Work Phase-II (18CSP83) entitled "Smart Traffic Management For Ambulance" is a bonafide work carried out by:

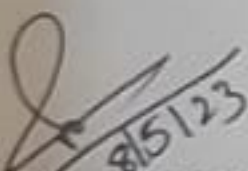
Sunil M (1KG19CS102)

V Yashaswini Naidu (1KG19CS108)

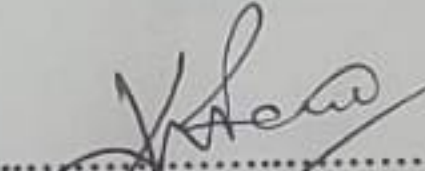
Vignesh R (1KG19CS115)

Vishwas P (1KG19CS117)

in partial fulfilment for VIII semester B.E., Project Work in the branch of Computer Science and Engineering prescribed by Visvesvaraya Technological University, Belagavi during the period of February 2023 to May 2023. It is certified that all the corrections and suggestions indicated for internal assessment have been incorporated. The Project Work Phase-2 Report has been approved as it satisfies the academic requirements in report of project work prescribed for the Bachelor of Engineering degree.

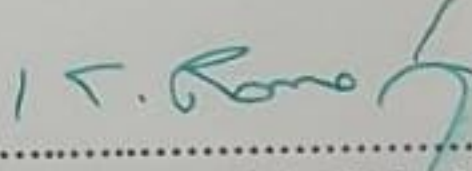

.....
Signature of the Guide

[Mrs, Amitha S]


.....
Signature of the HOD

[Dr. K Venkata Rao]

HOD
Department of Computer Science Engineering
K.S School of Engineering & Management
Bangalore-560109


.....
Signature of the Principal

[Dr. K Rama Narasimha]

Dr. K. RAMA NARASIMHA
Principal/Director
K S School of Engineering and Management
Bengaluru - 560 109

DECLARATION

We, the undersigned students of 8th semester, Computer Science & Engineering, KSSEM, declare that our Project Work Phase-II entitled "Smart Traffic Management For Ambulance", is a bonafide work of our's. Our project is neither a copy nor by means a modification of any other engineering project.

We also declare that this project was not entitled for submission to any other university in the past and shall remain the only submission made and will not be submitted by us to any other university in the future.

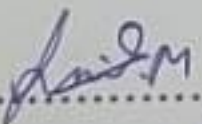
Place:

Date :

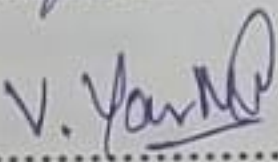
Name and USN

Signature

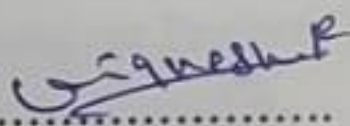
Sunil M (1KG19CS102)

.....

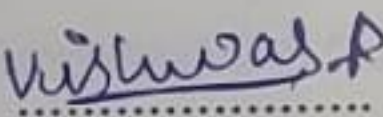
V Yashaswini Naidu (1KG19CS108)

.....

Vignesh R (1KG19CS115)

.....

Vishwas P (1KG19CS117)

.....

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task will be incomplete without the mention of the individuals, we are greatly indebted to, who through guidance and providing facilities have served as a beacon of light and crowned our efforts with success.

We would like to express our gratitude to our **MANAGEMENT**, K.S. School of Engineering and Management, Bengaluru, for providing a very good infrastructure and all the kindness forwarded to us in carrying out this project work in college.

We would like to express our gratitude to **Dr. K.V.A Balaji**, CEO, K.S. School of Engineering and Management, Bengaluru, for his valuable guidance.

We would like to express our gratitude to **Dr. K. Rama Narasimha**, Principal, K.S. School of Engineering and Management, Bengaluru, for his valuable guidance.

We like to extend our gratitude to **Dr. K Venkata Rao**, Professor and Head, Department of Computer Science & Engineering, for providing a very good facilities and all the support forwarded to us in carrying out this Project Work Phase-2 successfully.

We also like to thank our **Project Coordinators**, **Mrs. Nita Meshram** and **Mrs. Thejaswini M S** for their help and support provided to carry out the Project Work Phase-2 successfully.

Also, we are thankful to **Mrs. Amitha S** for being our **Project Guide**, under whose able guidance this project work has been carried out Project Work Phase-2 successfully.

We are also thankful to the teaching and non-teaching staff of Computer Science & Engineering, KSSEM for helping us in completing the Project Work Phase-2 work.

Name & USN

Sunil M (1KG19CS102)

V Yashaswini Naidu (1KG19CS108)

Vignesh R (1KG19CS115)

Vishwas P (1KG19CS117)

ABSTRACT

Smart traffic management for ambulances is an innovative solution aimed at improving the response time of emergency medical services (EMS) in highly congested urban areas. It is a system that utilizes advanced technologies to create a dedicated lane for ambulances, allowing them to bypass heavy traffic and reach their destination more quickly.

The smart traffic management system includes a network of sensors, cameras, and communication devices that work together to detect the presence of an ambulance in the vicinity and provide it with a clear path. This system is connected to a centralized traffic control centre that can modify traffic lights and provide traffic rerouting instructions to the drivers in real-time.

One of the primary benefits of this system is that it reduces the response time of EMS vehicles, which is crucial in situations where seconds can make a difference between life and death. By creating a dedicated lane for ambulances, the smart traffic management system ensures that they can reach their destination in the shortest possible time.

Another benefit of this system is that it reduces the workload of ambulance drivers, who often face high levels of stress and pressure when navigating through congested traffic. With the smart traffic management system, drivers can focus on providing medical care to their patients, knowing that they will not be delayed by traffic.

Moreover, this system also enhances the safety of ambulance drivers and patients by minimizing the risk of accidents caused by reckless driving or erratic behaviour of other drivers on the road. With a dedicated lane for ambulances, the risk of collisions and accidents is significantly reduced, ensuring the safety of everyone on the road.

In conclusion, smart traffic management for ambulances is an innovative solution that has the potential to revolutionize the way emergency medical services are provided in highly congested urban areas. By reducing response times, minimizing the workload of ambulance drivers, and enhancing safety, this system has the potential to save countless lives and make a significant contribution to the field of emergency medicine.

CONTENTS

<i>Acknowledgment</i>	
<i>Abstract</i>	I
<i>Contents</i>	II
<i>List of Figures</i>	V
<i>List of Tables</i>	VI

Chapter No.	Title	Page No.
1.	INTRODUCTION	1-6
1.1	Smart Traffic Management Concept	3
1.2	Problem Statement	3
1.3	Motivation/Need of the Project	4
1.4	System Study	4
1.5	Existing System	4
1.6	Proposed System	5
1.7	Scope of the Project	5
1.8	Objective of the Project	6
2.	LITERATURE SURVEY	7-12
2.1	An Automated Vehicle Counting System Based on Blob Analysis for Traffic Surveillance by G. Salvi.	7
2.2	Coordinated Road Junction Traffic Control by Dynamic Programming by Tsin Hing Heung	8
2.3	Intelligent Traffic Signalling System by Ch.Jaya Lakshmi S.Kalpna	8
2.4	Dynamic Traffic Management System Using Infrared Red(IR) and Internet of Things (IoT) by	9

Paul Jasmine Rani, Khoushik Kumar

2.5	Intelligent Cross Road Traffic Management System by Ahmed S. Salama	10
2.6	Outcome of Literature Survey	11
3.	SYSTEM REQUIREMENTS AND SPECIFICATION	13-15
3.1	General Description	13
3.2	Functional Requirements	14
3.3	Non-Functional Requirements	14
3.4	Hardware Requirements	15
3.5	Software Requirements	15
4.	SYSTEM DESIGN	16-24
4.1	Introduction	16
4.2	High Level Design	17
4.3	System Architecture	17
4.4	Data Flow Diagram	19
4.5	Use Case Diagram	21
4.6	Class Diagram	22
4.7	Sequence Diagram	23
5.	IMPLEMENTATION	25-40
5.1	Introduction	25
5.2	Components Description	26
5.3	Implementation Code	29
6.	TESTING	41-44
6.1	Test Environment	41

7.	RESULTS AND SNAPSHOTS	45
8.	APPLICATIONS	47
9.	CONCLUSION AND FUTURE SCOPE	48
9.1	Conclusion	48
9.2	Future Scope	48
10.	CONTRIBUTION TO SOCIETY	49
	REFERENCES	50
	APPENDIX-I Certificates of Paper Published	52-55
	APPENDIX-II Journal Paper Published	56-59

LIST OF FIGURES

Fig No.	Particulars	Page No.
1.1	Classification Of Ambulances	2
4.1	High Level Architectural design	18
4.2	System Architecture	18
4.3	Design Overview	19
4.4.1	DFD level 0	20
4.4.2	DFD level 1	20
4.4.3	DFD level 2	21
4.5	Use Case Diagram	22
4.6	Class Diagram	23
4.7	Sequence Diagram	24
5.2.1	Arduino	26
5.2.3(a)	Typical LED	27
5.2.3(b)	Circuit symbol	27
7.1	Detection Of Ambulance Using Image	45
7.2	Detection Of Ambulance Using Toys	45
7.2	Changing Of Signal	46

LIST OF TABLES

Table No.	Particulars	Page No.
6.1.1	Unit Test 1 Result	41
6.1.2	Unit Test 2 Result	42
6.1.3	Integration Test 1 Results	42
6.1.4	Integration Test 2 Results	43
6.1.5	System Test Results	43
6.1.6	Acceptance Test Results	44

CHAPTER – 1

INTRODUCTION

An ambulance is a specialized vehicle used to transport sick or injured people to medical facilities. The primary function of an ambulance is to provide emergency medical services, such as initial medical treatment, stabilization, and transport to a medical facility. Ambulances typically have a variety of medical equipment on board, such as oxygen supplies, defibrillators, first aid kits, and medications, to provide immediate medical care to patients during transportation. They may also be staffed by trained medical professionals, such as paramedics or emergency medical technicians (EMTs), who can provide advanced life support interventions as needed.

Ambulances can come in different types and sizes, depending on their specific purpose and the area they serve. For example, some ambulances may be designed for transporting critically ill patients, while others may be equipped for responding to mass-casualty incidents. Additionally, ambulances may be operated by private companies or government agencies, such as fire departments or emergency medical services (EMS).

In summary, an ambulance is a vital component of emergency medical services, providing rapid medical care and transportation to individuals who require immediate medical attention.

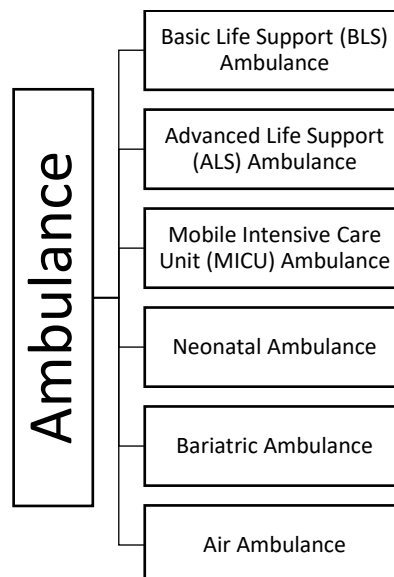


Figure 1.1: Classification of Ambulance Types

Fast transportation systems and rapid transit systems are nerves of economic developments for any nation. Mismanagement and traffic congestion results in long waiting times, loss of fuel and money. It is therefore utmost necessary to have a fast, economical and efficient traffic control system for national development. The monitoring and control of city traffic is becoming a major problem in many countries. With the ever increasing number of vehicles on the road, the Traffic Monitoring Authority has to find new methods of overcoming such a problem. One way to improve traffic flow and Safety of the current transportation system is to apply automation and Intelligent control methods. As the Number of road users constantly increases, and resources provided by current infrastructures are limited, intelligent control of traffic will become a very important issue in the future. Traffic congestion may result due to heavy traffic at a junction. To avoid congestion there are so many traffic management techniques available. But no technique is perfect by itself as the real time situations are generally continuously changing and the system has to adapt itself to change in the continuously changing circumstances. We have tried to provide some traffic management strategy which is self-changing in nature, so as to fit into continuously changing real time traffic scenarios. In this system time is assigned to traffic light of particular lane according to the traffic density on the road with priority given to ambulance. Also, we can indicate signal break in a particular lane. If there is an obstacle LCD is used to display the message of obstacle detection to avoid inconvenience.

The fact is that, the population of city and numbers of vehicles on the road are increasing day by day. With increasing urban population and hence the number of vehicles, need of controlling streets, highways and roads is major issue. The main reason behind today's traffic problem is the techniques that are used for traffic management. Today's traffic management system has no emphasis on live traffic scenario, which leads to inefficient traffic management systems. This project has been implemented by using the Python Open CV software and it aims to prevent heavy traffic congestion. Moreover, for implementing this project Image processing technique is used. At first, film of a lane is captured by a camera. Then these images are efficiently processed to know the traffic density. According to the processed data from mat lab, the controller will send the command to the traffic LEDs to show particular time on the signal to manage traffic.

This endless increasing number of vehicles on road gives rise to many problems amongst them traffic congestion tops in every aspect. In such scenario one cannot restrict individual to limit the usage of their private vehicles but what we can do is at least manage

traffic flow in a way that it doesn't alleviate congestion issues. There are many projects emerging in order to convert the current transport system of cities to 'Smart system' and there are many initiatives under this, one of this is Intelligent Transport System.

1.1 Smart Traffic Management Concept

Smart Traffic Management is an innovative approach to traffic management that uses advanced technologies and data analysis to improve traffic flow, reduce congestion, and enhance safety. The concept involves the integration of various technologies, such as sensors, cameras, GPS, and communication networks, to gather real-time traffic data and provide intelligent traffic management solutions.

Smart Traffic Management systems can utilize data analysis to identify congestion hotspots and adjust traffic signal timings accordingly to reduce traffic delays. It can also provide real-time traffic updates to drivers, enabling them to make informed decisions about their routes and avoid traffic congestion.

Additionally, Smart Traffic Management systems can enhance traffic safety by providing intelligent traffic management solutions, such as collision avoidance systems and speed limit alerts, to minimize the risk of accidents. It can also facilitate the smooth and safe passage of emergency vehicles, such as ambulances, through congested traffic, reducing response times for emergency services and increasing the safety of both patients and emergency personnel.

1.2 Problem Statement:

- Existing method used to control traffic is Traffic light System. A numerical value is loaded in timer for each phase. Depending on changes in timer, light is set at ON and OFF states on different lanes. Problem with this approach is that it is not good to have a green light on an empty lane.
- Another method used to control traffic is by using sensors. Sensors will get the traffic information about a particular lane and accordingly traffic lights might be set. But the problem is traffic information provided by sensors is limited.
- The main aim is to implement traffic management system which allocates the time to each lane based on the density and also to provide maximum priority to the lane where ambulance arrives.

1.3 Motivation/Need Of The Project:

The use of Image Processing and Embedded technology has proved to be very beneficial in present Traffic Light Controller (TLC) and that will minimize waiting time of emergency vehicles and also manage traffic load. In this project we exploit the emergence of new technology called as Intelligent traffic light controller, This makes the use of Segmentation of images along with embedded technology. Where traffic light will be intelligently decided based on the total traffic on all adjacent roads. Thus, optimization of traffic light switching increases road capacity, traffic flow and can prevent traffic congestions.

1.4 System Study

Robust and reliable traffic surveillance system is an urgent need to improve traffic control and management. Vehicle flow detection appears to be an important part in surveillance system. The traffic flow shows the traffic state in fixed time interval and helps to manage and control especially when there's a traffic jam. In this paper, we propose a traffic surveillance system for vehicle counting. The proposed algorithm is composed of five steps: background subtraction, blob detection, blob analysis, blob tracking and vehicle counting. A vehicle is modelled as a rectangular patch and classified via blob analysis. By analysing the blob of vehicles, the meaningful features are extracted. Tracking moving targets is achieved by comparing the extracted features and measuring the minimal distance between consecutive frames. The experimental results show that the proposed system can provide real-time and useful information for traffic surveillance.

A traffic police standing at junctions, or at cross roads, is the simplest and the oldest method used for the traffic management. It includes a human in the traffic system. A traffic officer is placed on each and every cross section of roads, and he manually controls the traffic. A police officer stands at middle of the road and monitoring the flow of traffic. The police officer gives signals to the vehicle driver whether to drive or start. And he always monitors every road, and decides which lane has to give first priority. Based on his own knowledge on his own takes the decisions, that which lane has to allow and which one to stop.

1.5 Existing System

The automatic traffic signal system include simple three colours: yellow, red and green. Normally 120 seconds of green light is on for each lane. The major problem with this

system is it cannot identify the amount of traffic in one particular lane. So, there is a chance of traffic jam. The existing system suffers from a limitation that to implement the project, the appropriate hardware has to be installed in every vehicle which can be comparatively difficult to install in two wheelers. Overall the entire framework is user dependent as the overall traffic congestion will depend on decision made by the user. The existing system has

- Manual method of traffic control by policemen
- Automatic timers
- Sensor based traffic control

1.6 Proposed System

Techniques proposed to control traffic signal by using image processing algorithms are:

- Image acquisition
- Image Enhancement
- Preprocessing
- Vehicle Density Measurement through IP and ML
- Traffic signal operation depending on the values received from Open CV Python

Advantages

- It avoids problems that usually arise with present traffic control systems.
- This Segmentation technique deals with a multi-vehicle, multilane, multi road junction area.
- It provides an efficient time management scheme, in which a dynamic time schedule is worked out in real time for the passage of each traffic column.

1.7 Scope Of The Project

- The population of city and numbers of vehicles on the road are increasing day by day.
- Traffic management system has no emphasis on live traffic scenario.
- To implement this project image processing technique is used.
- A camera is placed in a traffic lane that will capture image of road which we want to control traffic.

1.8 Objective Of The Project

To provide clear way to the ambulance whenever it enters into the range of camera and to control the signals by measuring the density of traffic thereby avoiding the wastage of time and saving the lives of human being.

CHAPTER – 2

LITERATURE SURVEY

2.1) An Automated Vehicle Counting System Based on Blob Analysis for Traffic Surveillance by G. Salvi.

Robust and reliable traffic surveillance system is an urgent need to improve traffic control and management. Vehicle flow detection appears to be an important part in surveillance system. The traffic flow shows the traffic state in fixed time interval and helps to manage and control especially when there's a traffic jam. In this paper, we propose a traffic surveillance system for vehicle counting. The proposed algorithm is composed of five steps: background subtraction, blob detection, blob analysis, blob tracking and vehicle counting. A vehicle is modeled as a rectangular patch and classified via blob analysis. By analyzing the blob of vehicles, the meaningful features are extracted. Tracking moving targets is achieved by comparing the extracted features and measuring the minimal distance between consecutive frames. The experimental results show that the proposed system can provide real-time and useful information for traffic surveillance. In recent years, the traffic surveillance system is put forward extensively to be studied because it can provide meaningful and useful information such as traffic flow density, the length of queue, average traffic speed and the total vehicle in fixed time interval. Generally, the traffic surveillance system requires more sensors. The common traffic sensors include (1) push button (detecting pedestrian demand), (2) loop detectors (detecting vehicle presence at one point), (3) magnetic sensors (magnometers), (4) radar sensors, (5) microwave detectors, (6) video cameras. The approach utilized to analyze traffic videos using following module pipeline: (1) background subtraction, (2) blob detection, (3) blob analysis, (4) blob tracking, (5) vehicle counting. In this paper the system has been implemented in C++. The system can process around 25 frames per second on a dual core processor at 2.4 GHz. Therefore, it presents a system to detect and count moving vehicles in traffic scenes. The virtual loop based method is used for the same. But the system is not able to recognize vehicle types. It could be used for classification for each detector in order to improve the statistic function. The results show that the proposed system can provide real-time and useful information for traffic surveillance which is further used to control traffic.

2.2) Coordinated Road Junction Traffic Control by Dynamic Programming by Tsin Hing Heung

This paper presents a novel approach to road-traffic control for interconnected junctions. With a local fuzzy-logic controller (FLC) installed at each junction, a dynamic-programming (DP) technique is proposed to derive the green time for each phase in a traffic-light cycle. Coordination parameters from the adjacent junctions are also taken into consideration so that organized control is extended beyond a single junction. Instead of pursuing the absolute optimization of traffic delay, this study examines a practical approach to enable the simple implementation of coordination among junctions, while attempting to reduce delays, if possible. The simulation results show that the delay per vehicle can be substantially reduced, particularly when the traffic demand reaches the junction capacity. When the population grows, traffic congestion only worsens, and constructing more roads does not always help matters much. In devising new measures or resources to accommodate the ever-increasing road-traffic demand, not much room is left for the city planners to maneuver. On the other hand, maximizing the capacity of the existing infrastructure remains one of the feasible means of relieving the congestion. In this paper, the author reviews the road-traffic control strategies and methodologies. A local fuzzy-logic controller (FLC) has been developed for individual junctions and it provides the basis for this coordinated control. To demonstrate the performance of this decentralized traffic control with DP junction coordination, it is applied on a simple two-junction network which is subject to various traffic demands, and the FLCs presented are adopted as the local junction controllers. The performance is compared with that attained by two local FLCs with fixed offsets only. The development of a generic decentralized traffic-junction controller is presented here. The controller is simple in structure and does not require any geographic knowledge of the road network. They employ fuzzy logic and genetic algorithms (GA) to handle the local control and the learning process, respectively.

2.3) Intelligent Traffic Signalling System by Ch.Jaya Lakshmi S.Kalpana

Traffic congestion has become a major problem in every large city of the world. To ensure a reliable transportation system it is important to have an intelligent traffic control system.

The very first step to do that is to acquire traffic data. Traffic data may be acquired from different methods. However, in recent days image processing techniques has been very important and promising topic to deal with traffic related problems because of its ease of maintenance and being more intelligent system. Most of the work detects edge of the vehicles and counts the number of traffic on the road. The disadvantage of the method is that counting the number of vehicles may give faulty results when space between the vehicles on the road are very small (i.e. two cars very close to each other may be counted as one vehicle). In this the proposed method firstly, compute the amount of area occupied by vehicles on the road rather than finding number of vehicles. The greater the amount of area occupied by vehicles on the road the greater the amount of traffic congestion. This way every kind of vehicles can be accounted for traffic density. Using this traffic data, the proposed model automatically controls the traffic signaling in a sequential manner depending on the amount of traffic on the road. In the proposed method, the amount of traffic on roads is estimated through measuring the total area occupied by vehicles on the road instead of vehicle count in terms of the traffic density. Variable traffic cycle is selected depending on the total traffic density of all the roads at the junction. Depending on the traffic density a weight is determined for each road and total traffic cycle is weighted for the roads. This way an intelligence based traffic signaling system may be designed. With the proposed system, the challenges involved in the traditional signaling system can be overcome. This model could be extended to incorporate a large number of interconnected traffic junctions and using their traffic density to adjust adjacent junction's time allocation. In this project, emergency conditions are not considered, so at that situation this smart traffic system fails and manual mechanism to be implemented. This problem can be overcome with some improvements.

2.4) Dynamic Traffic Management System Using Infrared Red(IR) and Internet of Things (IoT) by Paul Jasmine Rani, Khoushik Kumar

The major goal of the project is to make traffic management system work dynamically using Internet of Things, Infrared sensor and Image Processing in order to make traffic system work efficiently. Traffic management automation systems in the market aims to computerized the traffic lights, operates on a periodic schedule to control the light

(red/yellow/green) uses various technologies like GSM, NFC focuses on the basic operation of an electrical switch. Our project plan to provide an automated IR sense based solution that makes traffic signals to shift the lights (red/yellow/green) dynamically. We plan on implementing the project for one junction “Proof-of Concept” for this paper, which includes traffic lights, IR-sensors, Wi-Fi transmitter and Raspberry Pi microcontroller. The sensed data gathered from IR sensor is transmitted by the Wi-Fi transmitter which is received by the raspberry-pi controller. Based on this compilation it dynamically shifts time of the red signal and the user gets an intimation of status of the signal on his way. The Raspberry Pi controller works as a central console, it determines which sideways of the road signal is to get open or close. The central console gathers all the data from sensors and stores it in the cloud which intimates traffic status to a mobile device. The objective of our project is to construct an automated traffic management system, capable of distributing time on receiving signal from the Infrared sensors. Traffic management automation systems in the market aims to computerized the traffic lights, operates on a periodic schedule to shift the light (red/yellow/green) uses various technologies like GSM(Global system for Mobile Communication), NFC(Near Field Communication)etc., focuses on the basic operation of an electrical switch. By using this sort of system, there is a major demerit of waiting for long time in the signal. The side with less count of vehicles or no-vehicles is split the same time as the other sides with more crowded vehicles, hence paving way for long congestion near the signal area. On the other side, by using the method of dynamic traffic there is huge chance of getting this congestion to an end or making it simpler for the road users. The existing scheme faces a major demerit of changing the traffic controller in a clock-wise manner, it doesn't make note of the traffic denseness. The denseness of the traffic is calculated and the timer display is shift dynamically. This major advantage rules out the happening of ‘unwanted wait’ for the vehicles in the more crowded region.

2.5) Intelligent Cross Road Traffic Management System by Ahmed S. Salama

The aim of this research is to provide a design of an integrated intelligent system for management and controlling traffic lights based on distributed long range Photoelectric Sensors in distances prior to and after the traffic lights. The appropriate distances for sensors are chosen by the traffic management department so that they can monitor cars that are moving towards a specific traffic and then transfer this data to the intelligent software

that are installed in the traffic control cabinet, which can control the traffic lights according to the measures that the sensors have read, and applying a proposed algorithm based on the total calculated relative weight of each road. Accordingly, the system will open the traffic that are overcrowded and give it a longer time larger than the given time for other traffics that their measures proved that their traffic density is less. This system can be programmed with very important criteria that enable it to take decisions for intelligent automatic control of traffic lights. Also, the proposed system is designed to accept information about any emergency case through an active RFID based technology. Emergency cases such as the passing of presidents, ministries and ambulances vehicles that require immediate opening for the traffic automatically. As a result, the system will guarantee the fluency of traffic for such emergency cases or for the main vital streets and paths that require the fluent traffic all the time, without affecting the fluency of traffic generally at normal streets according to the time of the day and the traffic density. Also, the proposed system can be tuned to run automatically without any human intervention or can be tuned to allow human intervention at certain circumstances. In this paper an intelligent cross road traffic management and control system has been introduced. The proposed system aims to overcome the traffic jam, and create a flow of traffic in city streets through applying an intelligent algorithm based on calculating priorities that represent the total calculated relative weight of a specific direction in a cross road traffic. The system is capable of detecting the emergency cases vehicles such as ambulance cars through using a RFID technology or mobile device used by the driver and then providing a flow of traffic for such emergency cases. This proposed system needs to be applied first on experimental traffics, then to be generalized on all cities streets. In the future, the proposed system will be integrated with different types of transportation sources, including stationary and moving cameras, GPS (Geographic Positioning Systems) devices, historical databases and providing coherent and integrated information to users via a web-based interface.

2.6 Outcome of Literature Review

Based on my literature review, I found that a smart traffic management system for ambulances is a promising solution to improve emergency response times and increase the chances of survival for critically ill patients. The following are some of the key outcomes of the literature review:

1. Smart traffic management systems can reduce emergency response times by providing real-time traffic information to emergency medical services (EMS) personnel. This

information can help ambulance drivers avoid traffic congestion and take the fastest route to the hospital.

2. The use of intelligent transportation systems (ITS) can facilitate communication between EMS personnel and traffic control centres, allowing traffic signals to be adjusted to give priority to emergency vehicles.
3. GPS tracking and vehicle-to-vehicle communication can also be used to improve ambulance navigation and coordination. For example, ambulances can be equipped with sensors that communicate with traffic lights, allowing them to change to green as the ambulance approaches.
4. The integration of smart traffic management systems with electronic patient care records can also help EMS personnel identify the nearest hospital with the appropriate resources to treat a particular patient.
5. Several studies have shown that smart traffic management systems can significantly reduce emergency response times and improve patient outcomes. For example, a study conducted in New York City found that the use of ITS reduced ambulance travel times by 26%, resulting in an estimated 10% reduction in mortality for critically ill patients.

Overall, the literature suggests that a smart traffic management system for ambulances has the potential to significantly improve emergency response times and patient outcomes. However, more research is needed to evaluate the effectiveness of different smart traffic management strategies and to determine the optimal combination of technologies for different emergency response scenarios.

CHAPTER – 3

SYSTEM REQUIREMENTS AND SPECIFICATION

A requirement is some quality or performance demanded of a person in accordance with certain fixed regulations. A software requirements specification (SRS) – a requirements specification for a software system – is a complete description of the behaviour of a system to be developed. In addition to a description of the software functions, the SRS also contains non-functional requirements. Software requirements are a sub-field of software engineering that deals with the elicitation, analysis, specification, and validation of requirements for software.

3.1 General Description

Image processing is a form of processing for which input is an image or a series of images or videos such as photographs or frames of video with output as either an image or a set of characteristics or parameters related to the image. It is a method to perform some operations on an image in order to get an enhanced image or to extract some useful information from it. Image Processing includes following three steps:

- Importing image via image acquisition tools
- Analyzing and manipulating the image
- Output in which result can be an altered image or report that is based in image analysis.

There are two types of methods used for image processing namely analog and digital image processing.

Analog image processing can be used for hard copies like printouts and photographs. Image analysts use various fundamentals of interpretation while using this visual technique. Digital image processing techniques help in manipulation of the digital images by using computers. The three general phrases that all types of data have to undergo while using digital techniques are preprocessing, enhancement and display information extraction.

OPEN CV PYTHON is a general purpose programming language. It is used to process images by writing function files or script files to perform the operations. These files form a formal record of the processing used and ensures that the final result can be tested.

3.2 Functional Requirements

The functional requirements for a system describe what the system should do. These requirements depend on the type of software being developed, the general approach taken by the organization when writing requirements. The functional system requirements describe the system function in detail, its inputs and outputs, exceptions and so on.

- It provides the clear way for the ambulance to save the valued lives of people.
- Measurement of traffic density control.
- Easy flow of traffic.
- Greater efficiency of the System

3.3 Non- Functional Requirements

Non-functional requirements, as the name suggests, are requirements that are not directly concerned with the specific functions delivered by the system. They may relate to emergent system properties such as reliability, response time and store occupancy. Alternatively, they may define constraints on the system such as capabilities of I/O devices and the data representations used in system interfaces.

➤ Performance Requirements

The system is expected to have reasonable short time response. As the camera is continuously recording the video in traffic signal, once the ambulance is detected in fraction of second the way will be cleared immediately.

➤ Reliability

The system should be 99% reliable. Since it may need some maintenance or preparation for some particular day, the system does not need to be reliable every time. So, 80% reliability is enough.

➤ Efficiency

By changing the colour of single it will clear the way to the ambulance and saves lives.

➤ Availability

Camera, database, and neural network class classifier are always available any time.

➤ Maintainability

The system should be optimized for supportability, or ease of maintenance as far as possible.

3.4 Hardware Requirements

The most common type set of requirements defined by any operating system or software application is the physical computer resources also known as hardware. The hardware requirement list is:

- PC with open CV Python
- Web Camera
- Microcontroller-Arduino
- USB to UART
- LCD
- LED
- Buzzer

3.5 Software Requirements

The software requirements are description of features and functionalities of the system. The software requirement is a field within software engineering that deals with establishing the needs of stakeholders that are to be solved by software.

Requirements convey the expectations of users from the software product. The software requirement list is:

- Operating System : Windows 7 and Above
- Software Module : Open CV Image Processing.
- Interfacing : The interfacing between the hardware
- Computer : A general purpose PC as a central processing unit for various tasks

CHAPTER 4

SYSTEM DESIGN

Analysis is the process of breaking a complex topic or substance into smaller parts to gain a better understanding of it. Analysts in the field of engineering look at requirements, structures, mechanisms, and systems dimensions. Analysis is an exploratory activity. The Analysis Phase is where the project lifecycle begins. The Analysis Phase is where you break down the deliverables in the high-level Project Charter into the more detailed business requirements. The Analysis Phase is also the part of the project where you identify the overall direction that the project will take through the creation of the project strategy documents.

4.1 Introduction

Gathering requirements is the main attraction of the Analysis Phase. The process of gathering requirements is usually more than simply asking the users what they need and writing their answers down. Depending on the complexity of the application, the process for gathering requirements has a clearly defined process of its own. This process consists of a group of repeatable processes that utilize certain techniques to capture, document, communicate, and manage requirements.

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could see it as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering.

If the broader topic of product development "blends the perspective of marketing, design, and manufacturing into a single approach to product development," then design is the act of taking the marketing information and creating the design of the product to be manufactured. Systems design is therefore the process of defining and developing systems to satisfy specified requirements of the user.

System design is one of the most important phases of software development process. The purpose of the design is to plan the solution of a problem specified by the requirement documentation. In other words, the first step in the solution to the problem is the design of the project.

The design of the system is perhaps the most critical factor affecting the quality of the software. The objective of the design phase is to produce overall design of the software. It aims to figure out the modules that should be in the system to fulfil all the system requirements in an efficient manner.

The design will contain the specification of all these modules, their interaction with other modules and the desired output from each module. The output of the design process is a description of the software architecture.

The design phase is followed by two sub phases

- High Level Design
- Low Level Design

4.2 High Level Design

In the high-level design, the proposed functional and non-functional requirements of the software are depicted. Overall solution to the architecture is developed which can handle those needs. This chapter involves the following consideration.

- Design consideration
- Data flow diagram

4.2.1 Design Consideration

There are several design considerations issues that need to be addressed or resolved before getting down designing a complete solution for the system.

4.3 System Architecture

System architecture is a conceptual model that defines the structure and behaviour of the system. It comprises of the system components and the relationship describing how they work together to implement the overall system. The system works by detecting the entering objects to the scene, and tracking them throughout the video. The input to the algorithm is the raw video data of a site. The algorithm then performs the following steps: First, a statistical background model of the scene is populated using the first few frames of the video. This background model collects the statistics of the background of the recorded scene such as road, trees, buildings, etc. This model is then used to distinguish the objects

of interest (vehicles) from the surroundings. In the next step, the detected foreground parts of the scene are grouped together by a neighbourhood analysis, and a filtering process is applied to remove noise and misdetections.

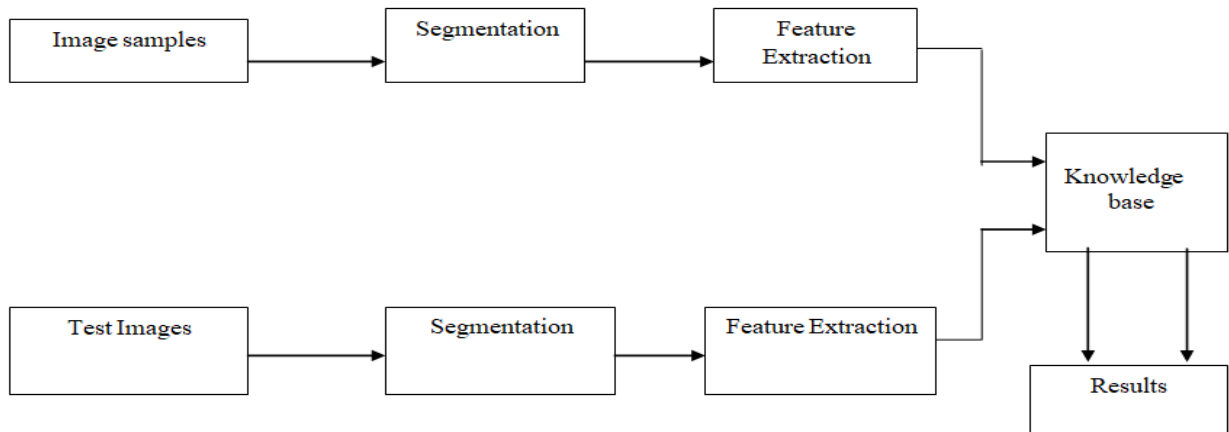


Figure 4.1 High Level Architectural design

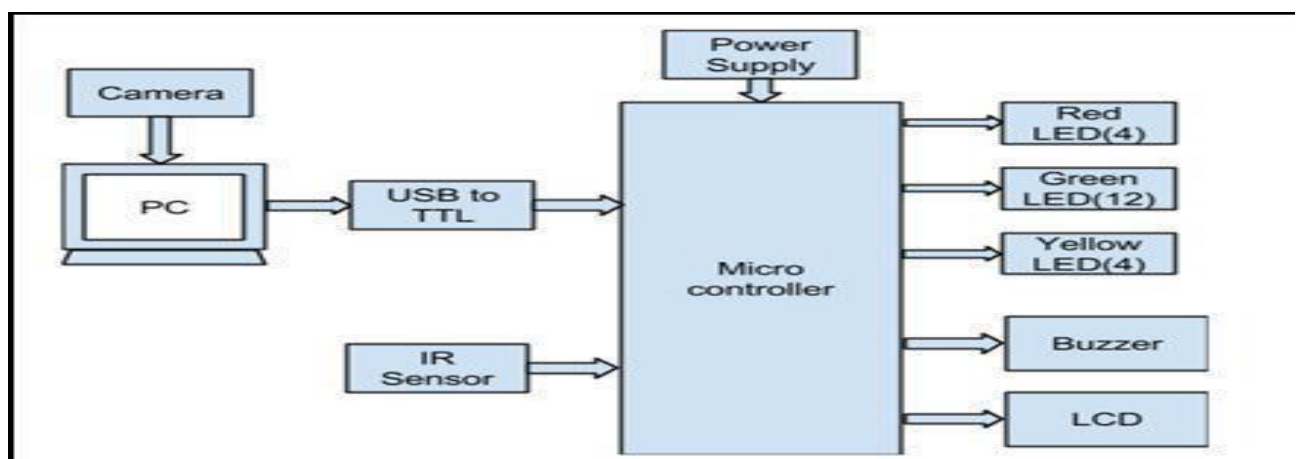


Figure 4.2 System Architecture

4.3.1 Data Collection

In this module datasets are collected from various sources like traffic at different places at different times. Several vehicles are considered for data collection. Cameras and sensors are used for the same.

4.3.2 Design Overview

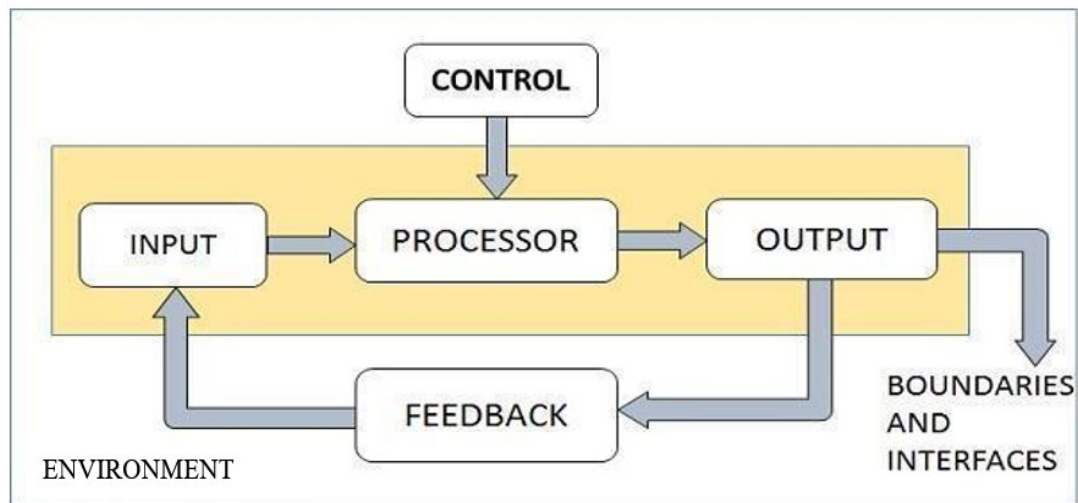


Figure 4.3 Design Overview

4.4 Data Flow Diagram

A data flow diagram is the graphical representation of the flow of data through an information system. DFD is very useful in understanding a system and can be efficiently used during analysis. A DFD shows the flow of data through a system. It views a system as a function that transforms the inputs into desired outputs. Any complex systems will not perform this transformation in a single step and a data will typically undergo a series of transformations before it becomes the output. With a data flow diagram, users are able to visualize how the system will operate that the system will accomplish and how the system will be implemented, old system data flow diagrams can be drawn up and compared with a new systems data flow diagram to draw comparisons to implement a more efficient system. Data flow diagrams can be used to provide the end user with a physical idea of where the data they input, ultimately as an effect upon the structure of the whole system.

4.4.1 Data Flow Diagram - Level 0

Basically, at zero level , we use processor and the ARDUINO processor. ARDUINO processor is present in the traffic signal itself. The processor is the software part of the project which are executing on the PC. ARDUINO processor acts as the hardware part where the input is received.

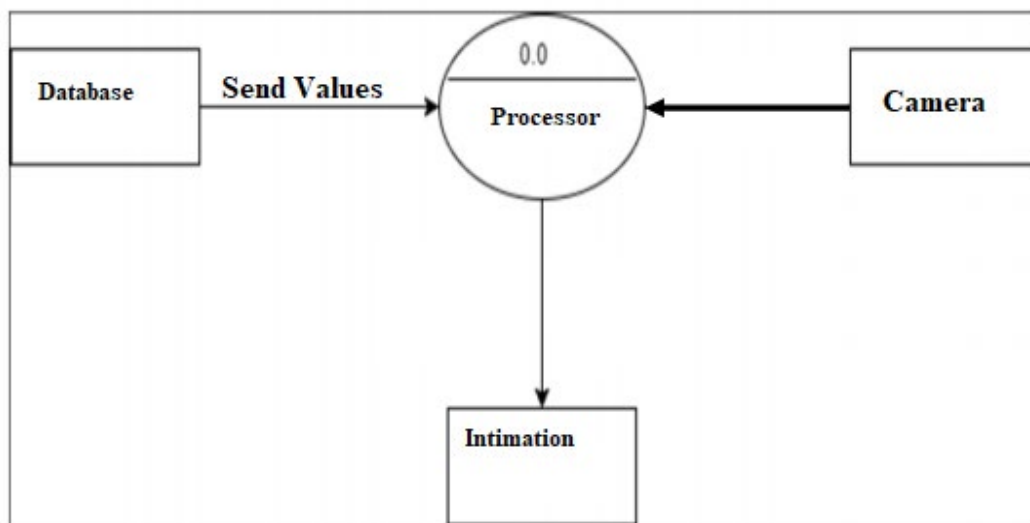


Figure 4.4.1 DFD level 0

4.4.2 Data Flow Diagram - Level 1

At level 1, comparison between reference image and captured image is done by processor. It processes and gives the input. ARDUINO processor receives the input from Open CV processor and allocates different time depending on the density of the signal.

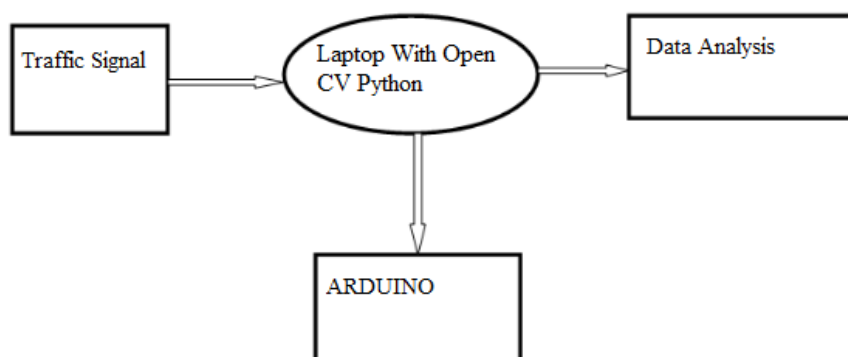


Figure 4.4.2 DFD level 1

4.4.3 Data Flow Diagram - Level 2

DFD Level 2 then goes one step deeper into parts of Level 1 of Traffic. It may require more functionalities of traffic to reach the necessary level of detail about the traffic functioning. First Level DFD of traffic monitoring shows how the system is divided into sub systems.

The 2nd level DFD contains more details of login ,Vehicle type, diversions ,traffic police, length, routes ,traffic. This level also includes image resizing ,rgb to gray conversion, image enhancement , image matching etc.

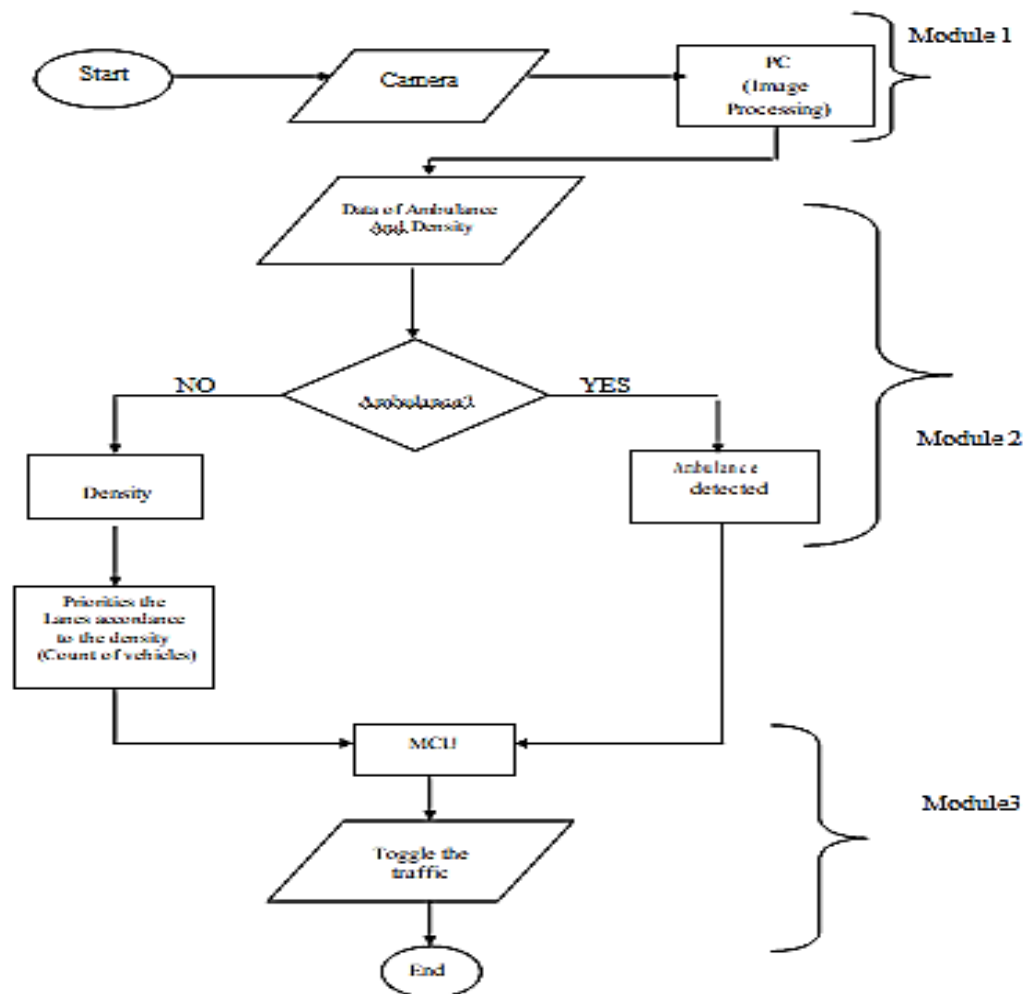


Figure 4.4.3 DFD level 2

4.5 Use Case Diagram

A use case diagram at its simplest 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 diagram 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. While a use case itself might drill into a lot of detail about every possibility, a use case diagram

can help provide a higher-level view of the system. It has been said before that "Use case diagrams are the blueprints for your system". They provide the simplified and graphical representation of what the system must actually do.

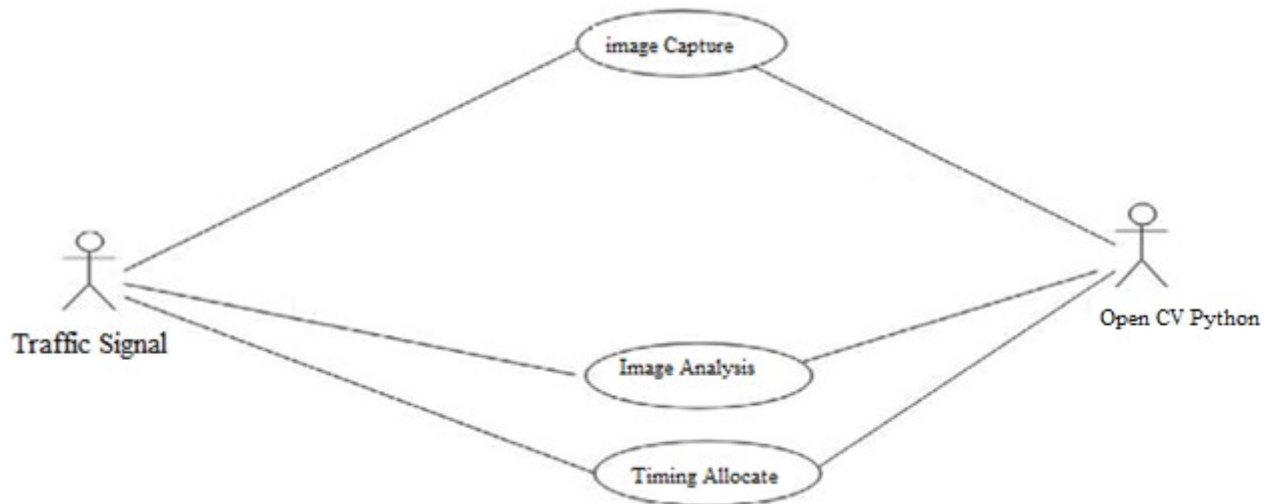


Figure 4.5 Use Case Diagram

4.6 Class Diagram

A class diagram in unified modified language is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operation, and the relationship among objects. The class diagram is the main building block of object-oriented modelling. It is used for general conceptual modelling of the structure of the application, and for detailed modelling translating the models into programming code.

Traffic control system describes the structure of traffic controller system classes, their attributes, operations and relationship among the objects. Classes of the traffic controller system class diagram are Traffic class, Route class, Length class, Traffic Police class, diversion class, Traffic light class.

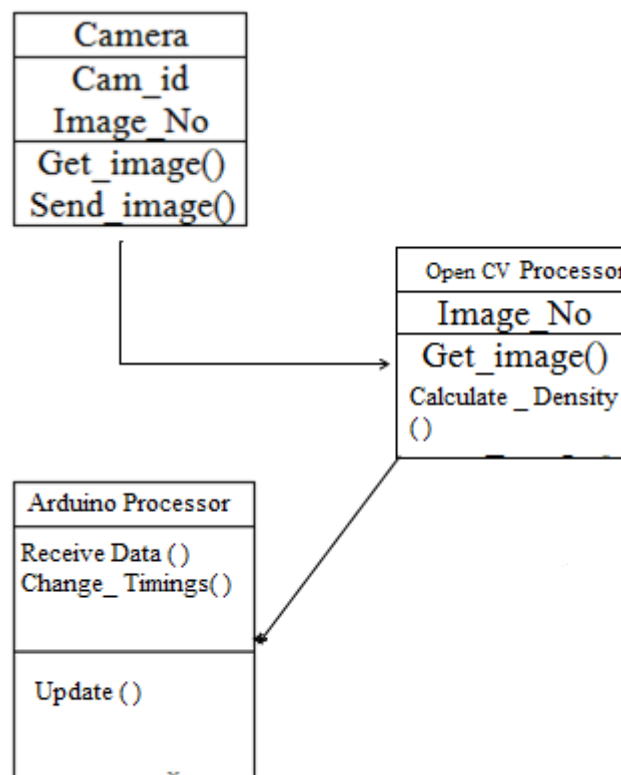


Figure 4.6 Class diagram

4.7 Sequence Diagram

A sequence diagram in a Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It shows the participants in an interaction and the sequence of messages among them; each participant is assigned a column in a table. Below section shows the sequence diagram in this application

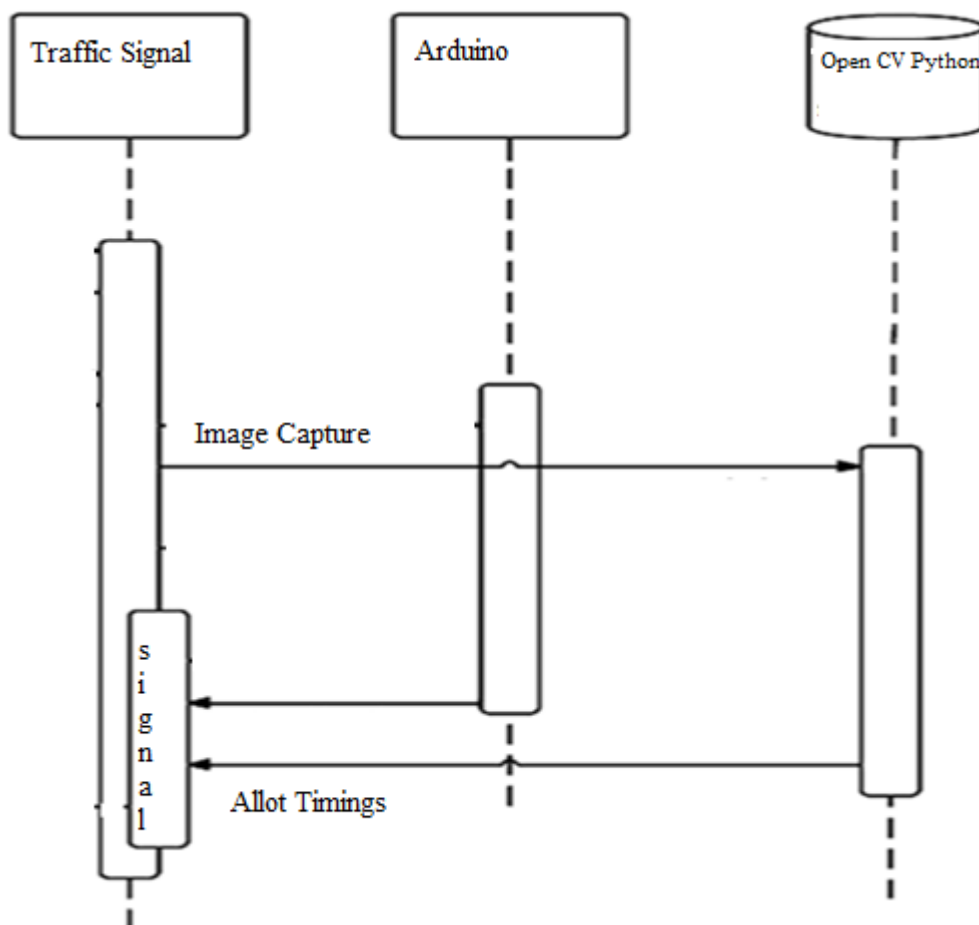


Figure 4.7 Sequence diagram

CHAPTER – 5

IMPLEMENTATION

5.1 Introduction

Implementation is the carrying out, execution or practice of a plan, a method, or any design, idea, model, specification, standard or policy for doing something. As such, implementation is the action that must follow any preliminary thinking in the order for something to actually happen. Implementations allow the users to take over its operation for use and evaluation. It involves training the users to handles the system and plan for a smooth conversion.

Implementation is a process of ensuring that the information system is

- Constructing a new system from scratch.
- Constructing a new system from the existing system.
- Traffic scene is overlooked by cameras at high posts of traffic lights. Extracted images from video are analyzed for detection and vehicles counting. After surveying we can reach to conclusion that Image processing is most efficient technique among all the existing methods in terms of efficiency, reliability, functionality, etc.

Here we propose a system called Intelligent Traffic Control using Image Processing, in which, vehicles are detected using cameras, which is placed along traffic light. An image is a rectangular graphical object. Smallest element of an image is a pixel; picture element, involves various issues regarding to image compression, enhancement techniques and various other operations which are complex. Such operations are for enhancement of images such as sharpening, blurring, brightening, etc. It is kind signal processing, in which, our image is the input signal. Output is an image or parameters of an image. Mostly, image is treated as two-dimensional signal. Signal processing techniques are applied to get the desired output. Image processing was mainly digital, but now analog Image Processing also came into existence. Cameras are used to capture images of roads which reflect traffic flow. Image Processing is beneficial and economical as cameras are affordable than electronic devices, sensors. We use a technique where captured image is compared to the referenced image, and various image processing operations are applied to get the result. For image morphology, finding optimal threshold value is very important.

5.2 Components Description

5.2.1 Arduino

The Arduino Mega is a microcontroller board based on the ATmega1280 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4UARTs (hardware serial ports), a 16MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to start.

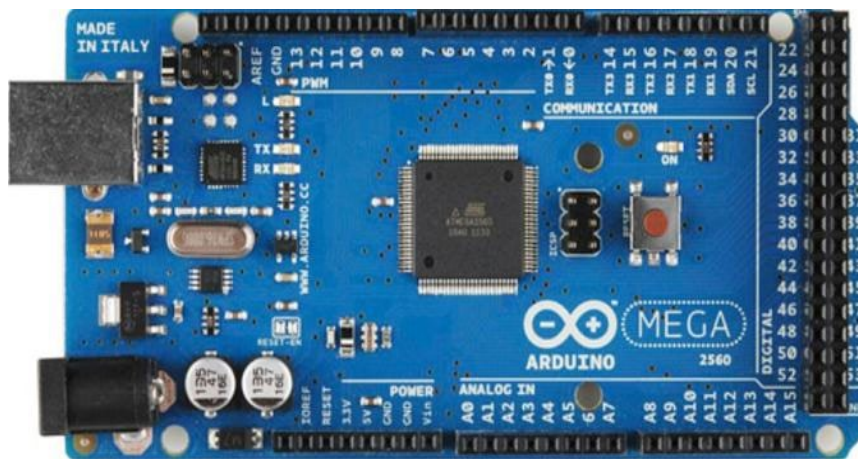


Figure 5.2.1 Arduino

The Arduino Mega 2560 can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm centre-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and VIN pin headers of the POWER connector.

5.2.2 Bus strips

To provide power to the electronic components .A bus strip usually contains two rows: one for ground and one for a supply voltage. However, some breadboards only provide a single-row power distributions bus strip on each long side.

Typically, the row intended for a supply voltage is marked in red, while the row for ground is marked in blue or black. Some manufacturers connect all terminals in a

column. Others just connect groups of, for example, 25 consecutive terminals in a column. The latter design provides a circuit designer with some more control over crosstalk on the power supply bus. Often the groups in a bus strip are indicated by gaps in the colour marking.

5.2.3 LED

LEDs are semiconductor devices. Like transistors, and other diodes, LEDs are made out of silicon. What makes an LED give off light are the small amounts of chemical impurities that are added to the silicon, such as gallium, arsenide, indium, and nitride. When current passes through the LED, it emits photons as a by-product. Normal light bulbs produce light by heating a metal filament until it is white hot. LEDs produce photons directly and not via heat, they are far more efficient than incandescent bulbs.

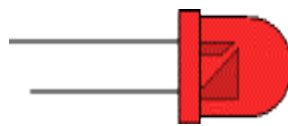


Figure 5.1.3(a): Typical LED

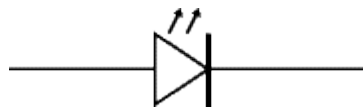


Figure 5.2.2(b): Circuit symbol

Not long ago LEDs were only bright enough to be used as indicators on dashboards or electronic equipment. But recent advances have made LEDs bright enough to rival traditional lighting technologies. Modern LEDs can replace incandescent bulbs in almost any application.

5.2.4 OPEN CV

Open CV is an open source C++ library for image processing and computer vision originally developed by Intel and now supported by Willow Garage. It is free for both commercial and non-commercial use. Therefore, it is not mandatory for your Open CV

communication to open for free it is a library of many inbuilt functions mainly aimed at real time image processing. Now it has several hundreds of image processing and computer vision algorithms which make developing advanced computer vision applications easy and efficient if you are having any troubles with installing Open CV or configure your Visual Studio IDE for Open CV, please refer to Installing and Configuring with Visual Studio.

OPENCV MODULES

Open CV has a modular structure. The main modules of Open CV are listed below. I have provided some links which are pointing to some example lessons under each module.

➤ Core

This is the basic module of Open CV. It includes basic data structures (e.g.- Mat data structure) and basic image processing functions. This module is also extensively used by other modules like highgui, etc.

➤ Highgui

This module provides simple user interface capabilities, several image and video codecs, image and video capturing capabilities, manipulating image windows, handling track bars and mouse events and etc. If you want more advanced UI capabilities,

➤ Imgproc

This module includes basic image processing algorithms including image filtering, image transformations, colour space conversions and etc.

➤ Video

This is a video analysis module which includes object tracking algorithms, background subtraction algorithms and etc.

➤ Objdetect

This includes object detection and recognition algorithms for standard objects. Open CV is now extensively used for developing advanced image processing and computer vision

applications. It has been a tool for students, engineers and researchers in every nook and corner of the world.

5.3 Implementation Code

5.3.1 Arduino Implementation Code

```
int red1=3;
int green1=2;
int red2=5;
int green2=4;
int red3=7;
int green3=6;
int red4=9;
int green4=8;

void setup() {
  Serial.begin(9600);
  pinMode(red1,OUTPUT);
  pinMode(green1,OUTPUT);
  pinMode(red2,OUTPUT);
  pinMode(green2,OUTPUT);
  pinMode(red3,OUTPUT);
  pinMode(green3,OUTPUT);
  pinMode(red4,OUTPUT);
  pinMode(green4,OUTPUT);
}

void loop() {
  initial_position();
}

void serial_event(){
  if(Serial.available() > 0){
```

```
String data = Serial.readString();
Serial.println(data);
char e = data[0];
Serial.println(e);
if(e == 'A'){
    road1(3);
}
else if(e == 'B'){
    road2(3);
}
else if(e == 'C'){
    road3(3);
}
else if(e == 'D'){
    road4(3);
}
else{
    int r1 = data[0]-48;
    int r2 = data[1]-48;
    int r3 = data[2]-48;
    int r4 = data[3]-48;

    int a, i, j, k, m;
    char b;
    int num[] = {r1, r2, r3, r4};
    char rd[] = "ABCD";

    for (i = 0; i < 4; ++i){
        for (j = i + 1; j < 4; ++j){
            if (num[i] < num[j]){
                a = num[i];
                b = rd[i];
                num[i] = num[j];
```



```
        rd[i] = rd[j];
        num[j] = a;
        rd[j] = b;
    }
}

for(k=0; k<4;k++){
    Serial.print(rd[k]);
    Serial.print(num[k]);
}
Serial.println("");
for(m=0; m<4;m++){
    if(rd[m] == 'A'){
        road1(num[m]);
    }
    if(rd[m] == 'B'){
        road2(num[m]);
    }
    if(rd[m] == 'C'){
        road3(num[m]);
    }
    if(rd[m] == 'D'){
        road4(num[m]);
    }
}
}
}

void initial_position(){
    road1(1);
    road2(1);
```

```
    road3(1);
    road4(1);
}
```

```
void road1(int C)
{
    digitalWrite(red1,LOW);
    digitalWrite(green1,HIGH);
    digitalWrite(red2,HIGH);
    digitalWrite(green2,LOW);
    digitalWrite(red3,HIGH);
    digitalWrite(green3,LOW);
    digitalWrite(red4,HIGH);
    digitalWrite(green4,LOW);
    delay(C*1000);
    serial_event();
}
```

```
void road2(int C)
{
    digitalWrite(red1,HIGH);
    digitalWrite(green1,LOW);
    digitalWrite(red2,LOW);
    digitalWrite(green2,HIGH);
    digitalWrite(red3,HIGH);
    digitalWrite(green3,LOW);
    digitalWrite(red4,HIGH);
    digitalWrite(green4,LOW);
    delay(C*1000);
    serial_event();
}
```

```
void road3(int C)
```

```
{  
    digitalWrite(red1,HIGH);  
    digitalWrite(green1,LOW);  
    digitalWrite(red2,HIGH);  
    digitalWrite(green2,LOW);  
    digitalWrite(red3,LOW);  
    digitalWrite(green3,HIGH);  
    digitalWrite(red4,HIGH);  
    digitalWrite(green4,LOW);  
    delay(C*1000);  
    serial_event();  
}
```

```
void road4(int C)  
{  
    digitalWrite(red1,HIGH);  
    digitalWrite(green1,LOW);  
    digitalWrite(red2,HIGH);  
    digitalWrite(green2,LOW);  
    digitalWrite(red3,HIGH);  
    digitalWrite(green3,LOW);  
    digitalWrite(red4,LOW);  
    digitalWrite(green4,HIGH);  
    delay(C*1000);  
    serial_event();  
}
```

5.3.2 Python Implementation Code

```
import argparse  
import os  
import platform  
import sys  
from pathlib import Path
```

```
import torch
import torch.backends.cudnn as cudnn

FILE = Path(__file__).resolve()
ROOT = FILE.parents[0] # YOLOv5 root directory
if str(ROOT) not in sys.path:
    sys.path.append(str(ROOT)) # add ROOT to PATH
ROOT = Path(os.path.relpath(ROOT, Path.cwd())) # relative

from models.common import DetectMultiBackend
from utils.dataloaders import IMG_FORMATS, VID_FORMATS, LoadImages,
LoadStreams
from utils.general import (LOGGER, check_file, check_img_size, check_imshow,
check_requirements, colorstr, cv2,
                        increment_path, non_max_suppression, print_args, scale_coords,
strip_optimizer, xyxy2xywh)
from utils.plots import Annotator, colors, save_one_box
from utils.torch_utils import select_device, time_sync
import time

from serial_test import send_data

@torch.no_grad()
def run(weights=ROOT / 'toys.pt', # model.pt path(s)
        source=ROOT / '1', # file/dir/URL/glob, 0 for webcam
        data=ROOT / 'data/coco128.yaml', # dataset.yaml path
        imgsz=(640, 640), # inference size (height, width)
        conf_thres=0.30, # confidence threshold
        iou_thres=0.45, # NMS IOU threshold
        max_det=1000, # maximum detections per image
        device="", # cuda device, i.e. 0 or 0,1,2,3 or cpu
        view_img=False, # show results
        save_txt=False, # save results to *.txt
        save_conf=False, # save confidences in --save-txt labels
        save_crop=False, # save cropped prediction boxes
        nosave=False, # do not save images/videos
        classes=None, # filter by class: --class 0, or --class 0 2 3
        agnostic_nms=False, # class-agnostic NMS
        augment=False, # augmented inference
        visualize=False, # visualize features
        update=False, # update all models
        project=ROOT / 'runs/detect', # save results to project/name
```

```
name='exp', # save results to project/name
exist_ok=False, # existing project/name ok, do not increment
line_thickness=3, # bounding box thickness (pixels)
hide_labels=False, # hide labels
hide_conf=False, # hide confidences
half=False, # use FP16 half-precision inference
dnn=False, # use OpenCV DNN for ONNX inference
):
source = str(source)
save_img = not nosave and not source.endswith('.txt') # save inference images
is_file = Path(source).suffix[1:] in (IMG_FORMATS + VID_FORMATS)
is_url = source.lower().startswith(('rtsp://', 'rtmp://', 'http://', 'https://'))
webcam = source.isnumeric() or source.endswith('.txt') or (is_url and not is_file)
if is_url and is_file:
    source = check_file(source) # download

# Directories
save_dir = increment_path(Path(project) / name, exist_ok=exist_ok) # increment run
(save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True, exist_ok=True) #
make dir

# Load model
device = select_device(device)
model = DetectMultiBackend(weights, device=device, dnn=dnn, data=data)
stride, names, pt, jit, onnx, engine = model.stride, model.names, model.pt, model.jit,
model.onnx, model.engine
imgsz = check_img_size(imgsz, s=stride) # check image size

print(names)

# Dataloader
if webcam:
    view_img = check_imshow()
    cudnn.benchmark = True # set True to speed up constant image size inference
    dataset = LoadStreams(source, img_size=imgsz, stride=stride, auto=pt)
    bs = len(dataset) # batch_size
else:
    dataset = LoadImages(source, img_size=imgsz, stride=stride, auto=pt)
    bs = 1 # batch_size
vid_path, vid_writer = [None] * bs, [None] * bs

# Run inference
model.warmup(imgsz=(1 if pt else bs, 3, *imgsz)) # warmup
```

```

dt, seen = [0.0, 0.0, 0.0], 0
global m
m = 1
global np
np = 1
count = 0
for path, im, im0s, vid_cap, s in dataset:
    t1 = time_sync()
    im = torch.from_numpy(im).to(device)
    im = im.half() if half else im.float() # uint8 to fp16/32
    im /= 255 # 0 - 255 to 0.0 - 1.0
    if len(im.shape) == 3:
        im = im[None] # expand for batch dim
    t2 = time_sync()
    dt[0] += t2 - t1

    # Inference
    visualize = increment_path(save_dir / Path(path).stem, mkdir=True) if visualize
else False
    pred = model(im, augment=augment, visualize=visualize)
    t3 = time_sync()
    dt[1] += t3 - t2

    # NMS
    pred = non_max_suppression(pred, conf_thres, iou_thres, classes, agnostic_nms,
max_det=max_det)
    dt[2] += time_sync() - t3

    # Second-stage classifier (optional)
    # pred = utils.general.apply_classifier(pred, classifier_model, im, im0s)
    road1=0
    road2=0
    road3=0
    road4=0
    counting=0
    # Process predictions
    for i, det in enumerate(pred): # per image
        seen += 1
        if webcam: # batch_size >= 1
            p, im0, frame = path[i], im0s[i].copy(), dataset.count
            s += f'{i}: '
        else:
            p, im0, frame = path, im0s.copy(), getattr(dataset, 'frame', 0)

```

```

p = Path(p) # to Path
save_path = str(save_dir / p.name) # im.jpg
txt_path = str(save_dir / 'labels' / p.stem) + (" if dataset.mode == 'image' else
f_{frame}') # im.txt
s += '%gx%g ' % im.shape[2:] # print string
gn = torch.tensor(im0.shape)[[1, 0, 1, 0]] # normalization gain whwh
imc = im0.copy() if save_crop else im0 # for save_crop
annotator = Annotator(im0, line_width=line_thickness, example=str(names))

height, width, _ = im0.shape

gridx1 = int(width/3)
gridx2 = gridx1 + gridx1

gridy1 = int(height/3)
gridy2 = gridy1 + gridy1

cv2.line(im0, (gridx1, 0), (gridx1, height), (0, 255, 0), 1)
cv2.line(im0, (gridx2, 0), (gridx2, height), (0, 255, 0), 1)
cv2.line(im0, (0, gridy1), (width, gridy1), (0, 255, 0), 1)
cv2.line(im0, (0, gridy2), (width, gridy2), (0, 255, 0), 1)

if len(det):
    # Rescale boxes from img_size to im0 size
    det[:, :4] = scale_coords(im.shape[2:], det[:, :4], im0.shape).round()

    # Print results
    for c in det[:, -1].unique():
        n = (det[:, -1] == c).sum() # detections per class
        s += f'{n} {names[int(c)]} {'s' * (n > 1)}, " # add to string

    # Write results
    for *xyxy, conf, cls in reversed(det):
        if save_txt: # Write to file
            xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-
1).tolist() # normalized xywh
            line = (cls, *xywh, conf) if save_conf else (cls, *xywh) # label format
            with open(txt_path + '.txt', 'a') as f:
                f.write('%g ' * len(line)).rstrip() % line + '\n')

        if save_img or save_crop or view_img: # Add bbox to image
            c = int(cls) # integer class

```

```

counting += 1
annotator.box_label(xyxy, str(counting), color=colors(c, True))

x, y, w, h = int(xyxy[0]), int(xyxy[1]), int(xyxy[2]), int(xyxy[3])

if x > gridx1 and x < gridx2 and y < gridy1:
    if names[c] == 'Ambulance':
        send_data('A')
        print('road1')
        road1 +=1

if x < gridx1 and y > gridy1 and y < gridy2:
    if names[c] == 'Ambulance':
        send_data('B')
        print('road2')
        road2 +=1

if x > gridx2 and y > gridy1 and y < gridy2:
    if names[c] == 'Ambulance':
        send_data('C')
        print('road4')
        road4 +=1

if x > gridx1 and x < gridx2 and y > gridy2:
    if names[c] == 'Ambulance':
        send_data('D')
        print('road3')
        road3 +=1

print('Total Vehicles in road1 {}'.format(road1))
print('Total Vehicles in road2 {}'.format(road2))
print('Total Vehicles in road3 {}'.format(road3))
print('Total Vehicles in road4 {}'.format(road4))
count1='{} {} {} {}'.format(road1, road2, road3, road4)
send_data(count1)

# Stream results
im0 = annotator.result()
cv2.imshow(str(p), im0)
cv2.waitKey(1) # 1 millisecond
fps, w, h = 50, im0.shape[1], im0.shape[0]
save_path = str(Path(save_path).with_suffix('.mp4')) # force *.mp4 suffix on
results videos

```



```

vid_writer[i] = cv2.VideoWriter(save_path, cv2.VideoWriter_fourcc(*'mp4v'),
fps, (w, h))
vid_writer[i].write(im0)

def parse_opt():
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', nargs='+', type=str, default=ROOT / 'toys.pt',
help='model path(s)')
    parser.add_argument('--source', type=str, default='0', help='file/dir/URL/glob, 0 for
webcam')
    parser.add_argument('--data', type=str, default=ROOT / 'data/coco128.yaml',
help='(optional) dataset.yaml path')
    parser.add_argument('--imgsz', '--img', '--img-size', nargs='+', type=int,
default=[640], help='inference size h,w')
    parser.add_argument('--conf-thres', type=float, default=0.25, help='confidence
threshold')
    parser.add_argument('--iou-thres', type=float, default=0.45, help='NMS IoU
threshold')
    parser.add_argument('--max-det', type=int, default=1000, help='maximum
detections per image')
    parser.add_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
    parser.add_argument('--view-img', action='store_true', help='show results')
    parser.add_argument('--save-txt', action='store_true', help='save results to *.txt')
    parser.add_argument('--save-conf', action='store_true', help='save confidences in --
save-txt labels')
    parser.add_argument('--save-crop', action='store_true', help='save cropped
prediction boxes')
    parser.add_argument('--nosave', action='store_true', help='do not save
images/videos')
    parser.add_argument('--classes', nargs='+', type=int, help='filter by class: --classes 0,
or --classes 0 2 3')
    parser.add_argument('--agnostic-nms', action='store_true', help='class-agnostic
NMS')
    parser.add_argument('--augment', action='store_true', help='augmented inference')
    parser.add_argument('--visualize', action='store_true', help='visualize features')
    parser.add_argument('--update', action='store_true', help='update all models')
    parser.add_argument('--project', default=ROOT / 'runs/detect', help='save results to
project/name')
    parser.add_argument('--name', default='exp', help='save results to project/name')
    parser.add_argument('--exist-ok', action='store_true', help='existing project/name ok,
do not increment')
    parser.add_argument('--line-thickness', default=2, type=int, help='bounding box
thickness (pixels)')

```

```
    parser.add_argument('--hide-labels', default=False, action='store_true', help='hide
labels')
    parser.add_argument('--hide-conf', default=False, action='store_true', help='hide
confidences')
    parser.add_argument('--half', action='store_true', help='use FP16 half-precision
inference')
    parser.add_argument('--dnn', action='store_true', help='use OpenCV DNN for
ONNX inference')
    opt = parser.parse_args()
    opt.imgsz *= 2 if len(opt.imgsz) == 1 else 1 # expand
    print_args(vars(opt))
    return opt

def main(opt):
    check_requirements(exclude=('tensorboard', 'thop'))
    run(**vars(opt))

if __name__ == "__main__":
    opt = parse_opt()
    main(opt)
```

CHAPTER – 6

TESTING

This chapter gives the outline of all testing methods that are carried out to get a bug free system. Quality can be achieved by testing the product using different techniques at different phases of the project development. The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components subassemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.1 Test Environment

Testing is an integral part of software development. Testing process certifies whether the product that is developed compiles with the standards that it was designed to. Testing process involves building of test cases against which the product has to be tested. write about the testing process used for the project.

Unit Testing

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. Unit testing is often automated but it can also be done manually. The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality. Test cases and results are shown in the Tables.

Table 6.1.1 Unit Test 1 Results

Sl # Test Case : -	UTC-1
Name of Test: -	Capture the Image
Items being tested: -	Tested for uploading different images
Sample Input: -	Upload Sample image
Expected output: -	Image should upload properly

Actual output: -	upload successful
Remarks: -	Pass.

Table 6.1.2 Unit Test 2 Results

Sl # Test Case : -	UTC-2
Name of Test: -	Density Calculation
Items being tested: -	Vehicles counting in captured image
Sample Input: -	Tested for different images with different no of vehicles
Expected output: -	Should count number of vehicles
Actual output: -	Should Display the count
Remarks: -	Pass

➤ Integration Testing:

Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing. Integration testing is defined as the testing of combined parts of an application to determine if they function correctly. It occurs after unit testing and before validation testing.

In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing. The process concludes with multiple tests of the complete application, preferably in scenarios designed to mimic actual situations. Table below shows the test cases for integration testing and their results.

Table 6.1.3 Integration Test 1 Results

Sl # Test Case : -	ITC-1
Name of Test: -	Working of Open CV with Hardware
Item being tested: -	Serial Communication between hardware and software
Sample Input: -	Click and select image

Expected output: -	Should send data to hardware serially
Actual output: -	Serial values should be received
Remarks: -	Pass.

Table 6.1.4 Integration Test 2 Results

Sl # Test Case : -	ITC-2
Name of Test: -	Signal Operation according to density
Item being tested: -	Place different number of vehicles and check
Sample Input: -	Capture image and send density count to hardware
Expected output: -	Signal LED's should operate according to density
Actual output: -	Traffic signal Status operated according to density
Remarks: -	Pass.

➤ **System testing:**

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic.

System Testing is shown in below tables

Table 6.1.5 System Testing Results

Sl # Test Case : -	STC-1
Name of Test: -	System testing in various versions of OS
Item being tested: -	OS compatibility.

Sample Input: -	Execute the program in windows XP/ Windows-7/8
Expected output: -	Performance is better in windows-7
Actual output: -	Same as expected output, performance is better in windows-7
Remarks: -	Pass

➤ Acceptance Testing

Acceptance testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Include recovery testing crashes, security testing for unauthorized user, etc.

Acceptance testing is sometimes performed with realistic data of the client to demonstrate that the software is working satisfactorily. This testing in FDAC focuses on the external behaviour of the system.

Table 6.1.6 Acceptance Testing Results

Test Case ID	System Test Case 1
Description	Intelligent Traffic Monitoring
Input	Threshold value
Expected output	Density Calculation
Actual Result/Remarks	Working as expected output.
Passed (?)	Yes

CHAPTER 7

RESULTS AND SNAPSHOTS

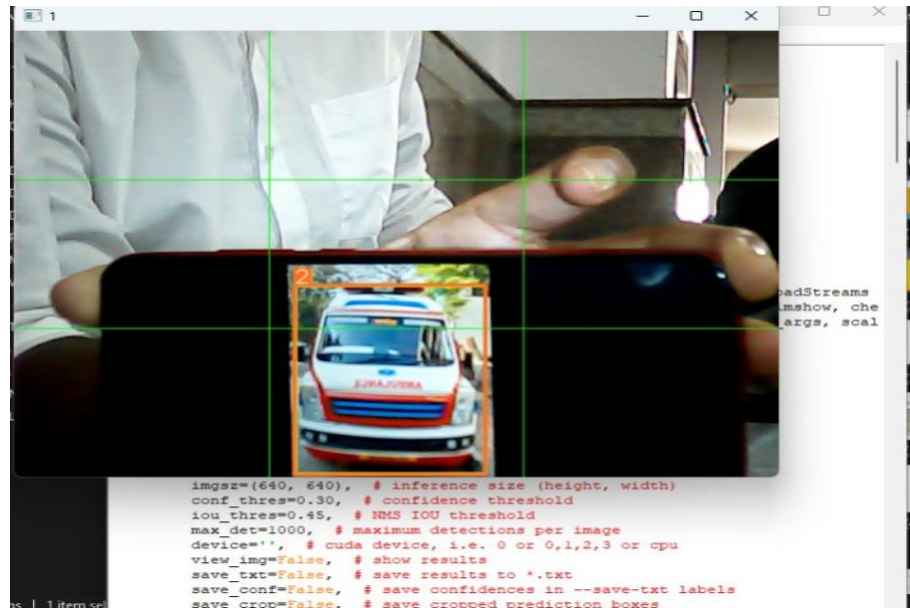


Figure 7.1 Detection Of Ambulance using image

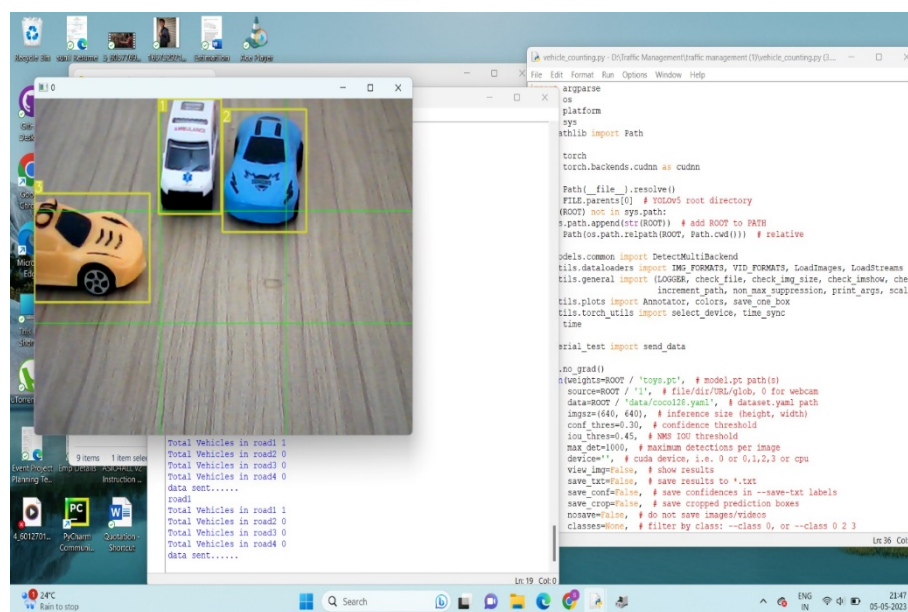


Figure 7.2 Detection Of Ambulance using Toys



Figure 7.3 Changing of Traffic Signal

CHAPTER 8

APPLICATIONS

Smart traffic management systems have revolutionized the way we manage traffic on our roads. With the increasing number of vehicles on the road, it has become essential to ensure that emergency vehicles such as ambulances can move quickly and efficiently through traffic. In this regard, smart traffic management systems can be particularly useful for ambulance services.

Here are some applications of smart traffic management for ambulance:

- Real-time traffic monitoring: Smart traffic management systems use real-time data to monitor traffic flow and congestion levels on the road. This can help ambulance services to identify the fastest and safest route to reach the patient in need.
- Traffic signal prioritization: Smart traffic management systems can prioritize traffic signals to give ambulances the right of way. This can significantly reduce the time taken for the ambulance to reach its destination, and also ensure that the patient receives timely medical attention.
- Incident management: Smart traffic management systems can be used to manage incidents on the road, such as accidents or roadblocks. This can help ambulance services to reroute their vehicles to avoid delays and reach the patient in need as quickly as possible.
- Automated emergency vehicle detection: Smart traffic management systems can detect emergency vehicles such as ambulances and automatically adjust traffic signals to clear the way for them. This can help ambulance services to avoid delays and reach their destination faster.
- Integration with other systems: Smart traffic management systems can be integrated with other systems, such as GPS and communication systems, to provide real-time information to ambulance services. This can help ambulance services to make informed decisions about the fastest and safest route to reach the patient in need.

CHAPTER 9

CONCLUSION AND FUTURE SCOPE

9.1 Conclusion

By this project the problem of traffic can be easily sorted out: the timing of each signal can be automatically adjusted according to density of traffic which is real time operation. It will also clear the path for the ambulance, fire brigade in emergency cases and also it will help to public in taking decisions for reaching their destination in time using auto-routing method. It shows that it can reduce the traffic congestion and avoids the time being wasted by a green light on an empty road. It is also more consistent in detecting vehicle presence because it uses actual traffic images. It visualizes the reality so it functions much better than those systems that rely on the detection of the vehicles metal content. Overall, the system is good but it still needs improvement to achieve a hundred percent accuracy.

The study showed that image processing is a better technique to control the state change of the traffic light. It shows that it can reduce the traffic congestion and avoids the time being wasted by a green light on an empty road. It is also more consistent in detecting vehicle presence because it uses actual traffic images. It visualizes the reality so it functions much better than those systems that rely on the detection of the vehicles' metal content. Overall, the system is good but it still needs improvement to achieve a hundred percent accuracy.

9.2 Future Enhancements

Going further, the details of the vehicle along with their RFID tag numbers can be stored on a centralized database. Cameras can be implemented in order to record driver's sight line, while GPS can also be installed to detect the present location of the vehicle.

In future we can extend with a camera, because if any person damages the light can be captured and sent to Server.

CHAPTER 10

CONTRIBUTION TO SOCIETY

Smart traffic management for ambulance is a technology that has significant contributions to society. One of the most significant contributions is the potential to save lives. In emergency situations, every second counts, and delays in getting ambulances to their destinations can have severe consequences. Smart traffic management systems can help ambulance services to navigate through traffic, reduce response times, and reach patients in need quickly.

Additionally, smart traffic management systems can help reduce traffic congestion and improve overall traffic flow. When an ambulance is on the road, other vehicles often have to yield or move out of the way, causing disruption to traffic flow. By prioritizing ambulance traffic and automatically adjusting traffic signals, smart traffic management systems can help reduce the disruption caused by emergency vehicles, while ensuring that they reach their destination quickly and efficiently.

Furthermore, smart traffic management systems can help reduce air pollution and carbon emissions. When ambulances are stuck in traffic, they often have to idle their engines, which contributes to air pollution. By reducing traffic congestion and improving traffic flow, smart traffic management systems can help reduce the amount of time that ambulances spend idling in traffic, and thus, reduce their environmental impact.

REFERENCES

- [1] Pandiaraj, K., et al. "RFID Based Automatic Lane Clearance for Ambulance." 2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE). IEEE, 2021.
- [2] Deepajothi, S., et al. "Intelligent Traffic Management for Emergency Vehicles using Convolutional Neural Network." 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS). Vol. 1. IEEE, 2021
- [3] Kamdar, Arihant, and Jigarkumar Shah. "Smart traffic system using traffic ow models." 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS). IEEE, 2021.
- [4] Rachana K P, Aravind R, Ranjitha M, Spoorthi Jwanita, Soumya K, 2021, IOT Based Smart Traffic Management System, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) NCCDS – 2021 (Volume 09 – Issue 12),
- [5] V. Bali, S. Mathur, V. Sharma and D. Gaur, "Smart Traffic Management System using IoT Enabled Technology," 2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), 2020, pp. 565-568, doi: 10.1109/ICACCCN51052.2020.9362753.
- [6] Lingani, Guy M., Danda B. Rawat, and Moses Garuba. "Smart traffic management system using deep learning for smart city applications." 2019 IEEE 9th annual computing and communication workshop and conference (CCWC). IEEE, 2019.
- [7] Javaid, Sabeen, et al. "Smart traffic management system using Internet of Things." 2018 20th international conference on advanced communication technology (ICACT). IEEE, 2018
- [8] Ghazal, Bilal, et al. "Smart traffic light control system." 2016 third international conference on electrical, electronics, computer engineering and their applications (EECEA). IEEE, 2016.

- [9] Lanke, Ninad, and Sheetal Koul. "Smart traffic management system." International Journal of Computer Applications 75.7 (2013)
- [10] Smart Traffic Control System for Emergency Vehicle Clearance, D.Aswani, Student, Department of ECE, C. Padma, (Ph.D), Assistant Professor, Department of ECE, Priyadarshini Institute of Technology, Ramachandrapuram, Tirupati, A.P., India

APPENDIX-I Certificates of Paper Published









APPENDIX-II Journal Published Paper



p-ISSN: 2582-5208

International Research Journal of Modernization in Engineering Technology and Science

(Peer-Reviewed, Open Access, Fully Refereed International Journal)

Volume:05/Issue:04/April-2023

Impact Factor- 7.868

www.irjmet.com

SMART TRAFFIC MANAGEMENT FOR AMBULANCE

Sunil M^{*1}, V Yashaswini Naidu^{*2}, Vignesh R^{*3},Vishwas P^{*4}, Amitha S^{*5}

^{*1,2,3,4}Dept. of Computer Science KS School Of Engineering and Management
Bangalore, India.

^{*5}Assistant Professor Dept. of Computer Science KS School Of Engineering and Management
Bangalore, India.

DOI : <https://www.doi.org/10.56726/IRJMET37427>

I. ABSTRACT

In emergency situations, every second counts, and timely medical assistance can mean the difference between life and death. However, with the increasing number of vehicles on the road and the ever-growing traffic congestion, ambulance response times are getting longer. To tackle this issue, a smart traffic management system for ambulance has been developed, which aims to optimize ambulance routes, reduce response times, and ultimately, save lives. The proposed system utilizes real-time data on traffic conditions, ambulance locations, and hospital availability to provide the most efficient and fastest route to the patient's location. The system also incorporates an intelligent traffic light control system that gives priority to ambulances, allowing them to pass through intersections quickly and safely. To evaluate the effectiveness of the system, a simulation study was conducted, which demonstrated that the smart traffic management system for ambulance can reduce response times by up to 50% compared to the traditional response system. Additionally, the system can handle multiple emergency cases simultaneously while still ensuring that the response times are optimized. In conclusion, the Smart Traffic Management System for Ambulance is a game-changer in emergency response, providing a faster and more efficient way of getting medical assistance to those in need. The impact of this system on emergency response times and ultimately on patient outcomes cannot be overstated, making it a crucial tool for emergency medical services in the future.

Keywords: Tracking, Ambulance, Traffic Jam.

II. INTRODUCTION

Ambulance response time is a crucial factor in the survival of patients during life-threatening situations. The time taken for an ambulance to reach a patient is directly proportional to the patient's chances of survival. However, traffic congestion is a significant barrier to reducing response times, particularly in urban areas. Smart traffic management systems have the potential to reduce these response times by providing ambulances with real-time traffic information and priority in traffic signals. This paper aims to review the current state-of-the-art technologies in smart traffic management for ambulances and explore future directions for research and development in this field.

III. IMPLEMENTATION OF THE SYSTEM

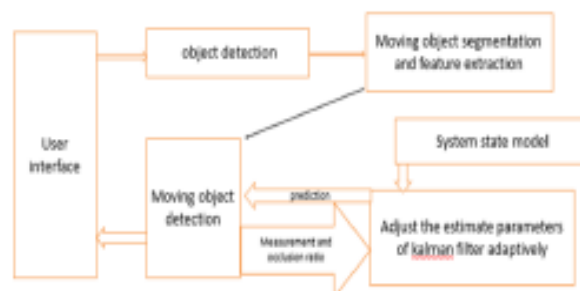


Fig. 1. Interface Design

www.irjmet.com

@International Research Journal of Modernization in Engineering Technology and Science

[6072]



e-ISSN: 2582-5208

International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)

Volume:05/Issue:04/April-2023 Impact Factor- 7.868

www.irjmet.com

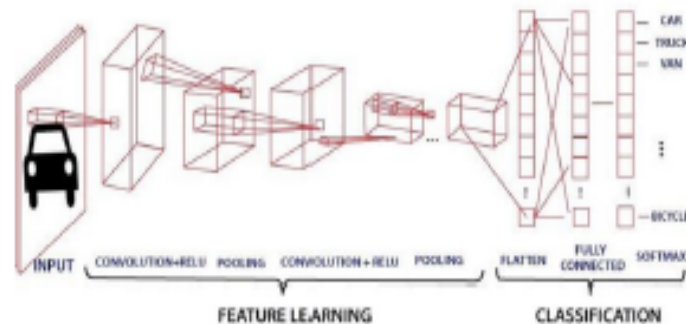


Fig. 2 Workflow

The CCTV's which is already functional set up by the traffic department for monitoring offences in traffic signals will be used for this project. The live video stream which is coming from this CCTV's is fed as input to the algorithm. The algorithm converts these videos into image frames and starts scanning for the ambulances. If the algorithm detects any image of ambulance which matches with the datasets given to the algorithm prior then it alerts the traffic signal receiver. The receiver at the traffic signal pole after receiving the alert turns the signal from red to green allowing the ambulance which is stuck in the traffic. This system is installed in all the traffic signals so that the ambulance reaches the destination without further stoppages.

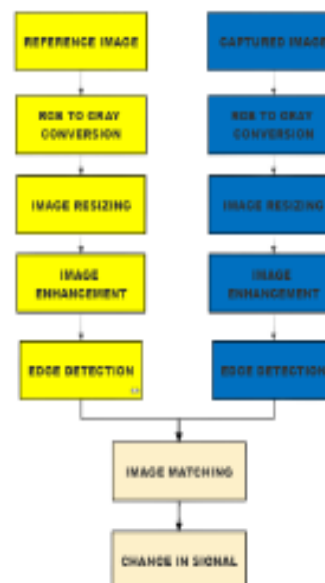


Fig. 3 Implementation of Flow chart

IV. COMPONENTS

The main components required for the functioning of the above proposed solution are elucidated below.

A. Arduino Uno

Arduino UNO is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started



e-ISSN: 2582-5208

International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)

Volume:05/Issue:04/April-2023

Impact Factor- 7.868

www.irjmet.com

B. Led Lights

5v led lights of green and red colour for indicating the traffic like signal on the model.

C. USB Camera

- Lens type : 3.6 mm
- Lens port : M12
- Closest focus distance : 0.7 m
- Power supply : 5V

V. RESULTS AND DISCUSSIONS

Emergency vehicle stuck in a traffic condition is detected with detection algorithm and then the traffic signal gets an alert by which it changes the signal to green from red. For the video part we use the already setup CCTV which is used by the traffic department for traffic violations. This live stream is broken into image frames and then the algorithm is run which searches with the dataset images and then alerts if a match is found from the image frames.

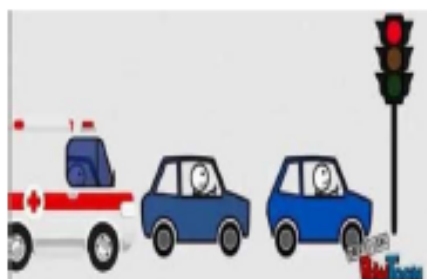


Fig. 5 Sketch of the System model

VI. APPLICATIONS

- It can be implemented in the city which has heavy traffic density.
- We can share the alert to nearby hospitals from the signal that an ambulance is coming and for the hospital staff to be ready.
- This technology can be implemented in any emergency cases like fire engine truck, VIP vehicles.

VII. CONCLUSION

In this paper, an idea is proposed for saving a patient's life in a faster way possible. It is beneficial for users in case of emergencies as it saves time. With this Application, the ambulance can reach the patients as fast as possible using the video and detection algorithm. This system makes sure that more lives are saved. It is easier for detecting ambulance and low cost. The difficulty is if the ambulance is stuck after a distance of more than 300 metres then it would be very difficult to detect the ambulance.

VIII. FUTURE ENHANCEMENT

This system can be implemented to all other emergency vehicles in future like fire engine, police, cars etc., in future. We can share the patient information to the hospital in an easy way. The patient information can be shared to the hospital so that they get ready with the required medicines or infrastructure to treat that patient. Traffic signal timer can be controlled dependent on number of vehicles present in front of the ambulance.

IX. REFERENCES

- [1] Pandiaraj, K., et al. "RFID Based Automatic Lane Clearance for Ambulance." 2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE). IEEE, 2021.
- [2] Deepajothi, S., et al. "Intelligent Traffic Management for Emergency Vehicles using Convolutional Neural Network." 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS). Vol. 1. IEEE, 2021.



e-ISSN: 2582-5208

International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)

Volume:05/Issue:04/April-2023

Impact Factor- 7.868

www.irjrets.com

- [3] Kamdar, Arhant, and Jigarkumar Shah. "Smart traffic system using traffic flow models." 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS). IEEE, 2021.
- [4] Rachana K P, Aravid R, Ranjitha M, Spoorthi Jwanita, Soumya K, 2021, IOT Based Smart Traffic Management System, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) NCCDS - 2021 (Volume 09 - Issue 12).
- [5] V. Bait, S. Mathur, V. Sharma and D. Gaur, "Smart Traffic Management System using IoT Enabled Technology," 2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), 2020, pp. 565-568, doi: 10.1109/ICACCCN51052.2020.9362753.
- [6] Lingani, Guy M., Danda B. Rawat, and Moses Garuba. "Smart traffic management system using deep learning for smart city applications." 2019 IEEE 9th annual computing and communication workshop and conference (CCWC). IEEE, 2019.
- [7] Javaid, Sabeen, et al. "Smart traffic management system using Internet of Things." 2018 20th International conference on advanced communication technology (ICACT). IEEE, 2018.
- [8] Ghazal, Bilal, et al. "Smart traffic light control system." 2016 third international conference on electrical, electronics, computer engineering and their applications (EECEA). IEEE, 2016.
- [9] Lanke, Ninad, and Sheetal Koul. "Smart traffic management system." International Journal of Computer Applications 75.7 (2013).
- [10] Smart Traffic Control System for Emergency Vehicle Clearance, D.Aswani, Student, Department of ECE, C. Padma, (Ph.D), Assistant Professor, Department of ECE, Priyadarshini Institute of Technology, Ramachandrapuram, Tirupati, A.P., India.