DrillBit

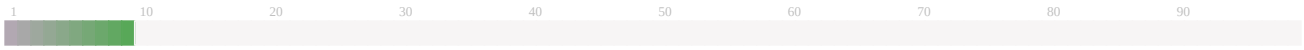
The Report is Generated by DrillBit Plagiarism Detection Software

### *Submission Information*

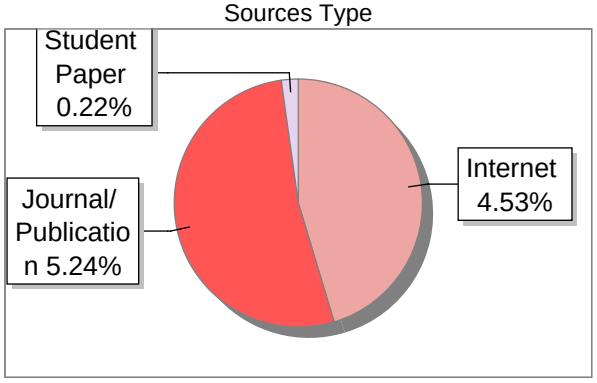
| | |
|---|---|
| Author Name | Komal S Kallanagoudar |
| Title | SmartFlow: Prioritize Emergency Vehicle and Traffic Rules Monitoring System |
| Paper/Submission ID | 2044067 |
| Submitted by | chieflibrarian@msrit.edu |
| Submission Date | 2024-06-24 13:12:04 |
| Total Pages, Total Words | 38, 8454 |
| Document type | Project Work |

### *Result Information*

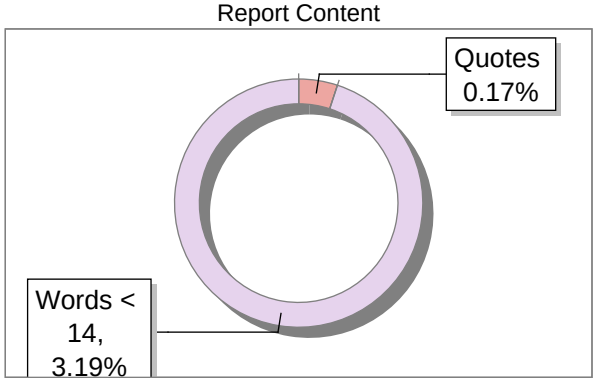
Similarity **10 %**




Sources Type




Report Content




### *Exclude Information*

| | | | |
|---|---|---|---|
| Quotes | Not Excluded | Language | English |
| References/Bibliography | Not Excluded | Student Papers | Yes |
| Source: Excluded < 14 Words | Not Excluded | Journals & publishers | Yes |
| Excluded Source | **0 %** | Internet or Web | Yes |
| Excluded Phrases | Excluded | Institution Repository | No |

### *Database Selection*

A Unique QR Code use to View/Download/Share Pdf File

DrillBit

**DrillBit Similarity Report**

| 10 | 41 | A |
|---|---|---|
| SIMILARITY % | MATCHED SOURCES | GRADE |

**A-Satisfactory (0-10%)**
**B-Upgrade (11-40%)**
**C-Poor (41-60%)**
**D-Unacceptable (61-100%)**

| LOCATION | MATCHED DOMAIN | % | SOURCE TYPE |
|---|---|---|---|
| 1 | fastercapital.com | 1 | Internet Data |
| 2 | Institute of Electrical and Electronics Engineers Article | 1 | Publication |
| 3 | vsit.edu.in | 1 | Publication |
| 4 | www.readbag.com | 1 | Internet Data |
| 5 | docplayer.net | <1 | Internet Data |
| 6 | www.jetir.org | <1 | Publication |
| 7 | ijrte.org | <1 | Publication |
| 8 | moam.info | <1 | Internet Data |
| 9 | www.linkedin.com | <1 | Internet Data |
| 10 | ijartet.com | <1 | Publication |
| 11 | www.ijert.org | <1 | Internet Data |
| 12 | www.learntek.org | <1 | Internet Data |
| 13 | IEEE 2019 International Conference on Machine Learning, Big Data, Cl | <1 | Publication |
| 14 | Institute of Electrical and Electronics Engineers Article | <1 | Publication |

| | | | |
|---|---|---|---|
| 15 | siem.sandipfoundation.org | <1 | Publication |
| 16 | Submitted to Visvesvaraya Technological University, Belagavi | <1 | Student Paper |
| 17 | Thesis submitted to dspace.mit.edu | <1 | Publication |
| 18 | abnews-wire.blogspot.com | <1 | Internet Data |
| 19 | digitalcommons.unomaha.edu | <1 | Publication |
| 20 | www.mdpi.com | <1 | Internet Data |
| 21 | documents.mx | <1 | Internet Data |
| 22 | www.ijirmf.com | <1 | Publication |
| 23 | gscl.assam.gov.in | <1 | Publication |
| 24 | Unsupervised Summarization and Change Detection in High-Resolution Signalized In by Mahajan-2020 | <1 | Publication |
| 25 | A trace cache microarchitecture and evaluation by Rotenberg-1999 | <1 | Publication |
| 26 | abnews-wire.blogspot.com | <1 | Internet Data |
| 27 | arpskakching.globizsdemo.com | <1 | Internet Data |
| 28 | bohrpub.com | <1 | Internet Data |
| 29 | County-Level Effects of Green Space Access on Physical Activity by Coutts-2013 | <1 | Publication |
| 30 | umpir.ump.edu.my | <1 | Internet Data |
| 31 | A Priority Queue with General Bulk Service by Gupta-1982 | <1 | Publication |
| 32 | frontline.thehindu.com | <1 | Internet Data |
| 33 | moam.info | <1 | Internet Data |

| | | | |
|---|---|---|---|
| 34 | Multiobjective Optimization for Multimodal Evacuation by Abdelgawad-2010 | <1 | Publication |
| 35 | Quantification of fructan concentration in grasses using NIR spectrosc by Nish-2011 | <1 | Publication |
| 36 | technodocbox.com | <1 | Internet Data |
| 37 | www.dx.doi.org | <1 | Publication |
| 38 | www.dx.doi.org | <1 | Publication |
| 39 | www.ijedr.org | <1 | Publication |
| 40 | www.linkedin.com | <1 | Internet Data |
| 41 | www.wto.org | <1 | Publication |

## EXCLUDED PHRASES

| | |
|---|---|
| 1 | **m s ramaiah institute of technology** |
| 2 | **msrit** |
| 3 | **bangalore** |

# 1. Introduction

## 1.1.Overview

SmartFlow is an innovative system that enhances road safety and traffic management by combining advanced technologies for emergency vehicle detection and real-time traffic rule monitoring. The project addresses the critical need for efficient emergency response and improved adherence to traffic regulations. Through the integration of cutting-edge sensors, computer vision, and artificial intelligence, SmartFlow aims to create a smarter and safer traffic environment.

### Objectives:

- Strengthen traffic monitoring efforts with camera technology.
- Controlling the traffic signals at dynamically based on real-time traffic data.
- Adaptive traffic signal systems adjust signal timings to minimize waiting times and reduce idling.
- The emergency vehicle is detected, which gives ambulances priority to pass through traffic lights.

## 1.2.Feasibility study

Traffic management systems in metropolitan areas would focus on evaluating the technical, economic, operational, legal, environmental, and social aspects of implementing advanced solutions. From a technical perspective, the study would assess the integration of camera-based monitoring, IoT sensors, and AI algorithms into existing infrastructure, ensuring compatibility and scalability across a complex urban network. Legally, regulatory compliance regarding data privacy, traffic management laws, and public safety would be scrutinized. Environmental impact assessments would focus on benefits such as reduced emissions and improved air quality, aligning with sustainability goals. Social feasibility would address public acceptance, community engagement, and equitable distribution of benefits, aiming to mitigate concerns about privacy and social equity.

## 1.3.Existing system

The existing traffic management systems in metropolitan cities face significant challenges due to the rapid growth in population and vehicle numbers. Traditional methods, such as manual traffic control by police officers and static traffic light systems, struggle to cope with the dynamic and increasing traffic demands. This inefficiency leads to frequent congestion, longer commute times, and heightened environmental pollution from vehicle emissions.

Emergency vehicles, such as ambulances and fire trucks, often find themselves stuck in traffic, unable to navigate swiftly through crowded intersections. This delay in emergency response can have severe consequences, including compromised public safety and increased risk to lives. The lack of real-time monitoring also means that traffic incidents and violations are often detected too late, resulting in delayed interventions and prolonged traffic disruptions.

## 1.4.Proposed system

The challenge is to design a SmartFlow System integrating camera technology for traffic monitoring and efficient prioritization of emergency vehicles through traffic lights.

The traffic management system of a metropolitan city is a keystone for urban mobility. With the rise of the population, the demand for vehicles grows up and hence the requirement of transportation has also increased. Infrastructural development becomes an indispensable part of complementing the population growth to augment urban mobility. But the traditional traffic management system is shown not only ineffective for accompanying the increased number of vehicles with the use of police control and traffic light system but also incompetent enough to handle this growth of traffic on road systems. This traffic congestion consequentially consumes precious working time and eventually leads to the environmental pollution for an extended period of vehicle emission. Traffic system utilize the concept of automation with IoT is called as “SmartFlow”. SmartFlow Management System is an advanced and integrated solution designed to optimize traffic flow, reduce congestion, enhance road safety, and improve overall transportation efficiency within urban or metropolitan areas.

# 2. Hardware and Software Requirements

## Hardware Requirements

- Arduino MEGA
- Arduino Uno
- ESP32-cam
- Traffic LEDs
- Jumper wires

## Software Requirements

- Arduino IDE
- Python IDLE
- PyCharm
- Xampp database

# 3. Software Requirements Specification

## 4.1.Overall Description

### 4.1.1. Product Perspective

The "SmartFlow: Prioritize Emergency Vehicle and Traffic Rules Monitoring System" project was conceived to tackle the critical challenge of ensuring the swift and safe passage of emergency vehicles amidst growing urban traffic congestion while maintaining strict adherence to traffic regulations. This innovative system is a new, self-contained product specifically designed to address the urgent need for efficient emergency response times and improved public safety.

The increasing density of urban traffic has made it imperative to develop systems that can facilitate the rapid movement of emergency vehicles such as ambulances, fire trucks, and police cars. Delays in emergency response can have dire consequences, making it essential to prioritize these vehicles through intelligent traffic management. SmartFlow is developed to address these issues by integrating modern technologies like IoT, GPS, and real-time data analytics.

SmartFlow utilizes a combination of IoT sensors and real-time data analytics to monitor and manage traffic. An ESP32-camera is employed to capture images of traffic conditions, which are then analyzed to determine congestion levels and identify the presence of emergency vehicles. When an emergency vehicle is detected, a signal is sent to an Arduino Mega, which adjusts the traffic signals to provide a green corridor for the vehicle.

### 4.1.2. Product Functions

- Image Capturing
- Vehicle Detection and counting
- Emergency Vehicle Detection
- Congestion Control
- Calculating green delay
- Update green signal

### 4.1.3. User Classes and Characteristics

The SmartFlowt system caters to various user classes with distinct characteristics. These user classes play specific roles and interact with the system differently. Here are some key user classes and their characteristics in a smart traffic management system:

- **Traffic Authorities:**
  Characteristics: Government agencies, traffic control centers, and law enforcement. Roles: Monitor and control traffic flow, receive real-time data for decision-making, and enforce traffic regulations.
  Interactions: Access to comprehensive traffic data, control traffic signals, and receive alerts for emergencies.
- **Transportation Planners:**
  Characteristics: Urban planners, city officials, and transportation departments.
  Roles: Analyze traffic patterns, plan infrastructure developments, and optimize transportation systems.
  Interactions: Access historical and real-time traffic data for city planning, implement changes based on insights, and assess the impact of transportation policies.
- **Individual Drivers:**
  Characteristics: Private vehicle owners and drivers.
  Roles: Navigate efficiently, avoid traffic congestion, and receive real-time information about road conditions.
  Interactions: Use navigation apps for real-time traffic updates, receive alerts on alternative routes, and contribute data through connected devices.
- **Pedestrians and Cyclists:**
  Characteristics: Individuals using non-motorized transportation modes.
  Roles: Ensure safe crossings, access pedestrian-friendly routes, and receive alerts on potential hazards.
  Interactions: Receive real-time information on pedestrian-friendly routes, crossing times, and safety alerts through mobile apps or connected devices.

- **IoT Device Manufacturers:**
  Characteristics: Companies producing IoT devices and sensors for traffic management. Roles: Develop and provide sensors for data collection, contribute to technology advancements, and improve system connectivity.
  Interactions: Collaborate with system administrators to ensure compatibility, offer updates for IoT devices, and participate in system improvement initiatives.
- **Emergency Services:**
  Characteristics: Firefighters, paramedics, and law enforcement.
  Roles: Respond to emergencies, navigate through traffic efficiently, and coordinate responses.
  Interactions: Access real-time traffic data for emergency route planning, receive priority signals at intersections, and collaborate with traffic authorities for incident management.

### 4.1.4. Operating Environment

SmartFlow operates within an urban environment, leveraging a diverse array of hardware platforms, operating systems, and software components. The primary hardware includes IoT sensors such as the ESP32-Camera for capturing real-time traffic images and Arduino Mega for processing signals and controlling traffic lights. These devices are essential for monitoring congestion levels and detecting emergency vehicles. The software runs on embedded systems programmed using the Arduino IDE and custom firmware designed for real-time processing. Python is used extensively for developing data analytics algorithms, while OpenCV handles image processing tasks. Databases like MySQL or PostgreSQL store captured images, traffic data, and system logs, ensuring reliable data management.

SmartFlow integrates seamlessly with existing city-wide traffic management systems, ensuring compatibility and smooth operation. It also shares real-time data with emergency response systems and law enforcement databases to monitor traffic conditions, prioritize emergency vehicles, and enforce traffic regulations. This integration enhances emergency response efficiency and improves urban mobility.

### 4.1.5. Design and Implementation Constraints

SmartFlow operates within an urban environment, leveraging a diverse array of hardware platforms, operating systems, and software components. The primary hardware includes IoT sensors such as the ESP32-Camera for capturing real-time traffic images and Arduino Mega for processing signals and controlling traffic lights. These devices are essential for monitoring congestion levels and detecting emergency vehicles. The software runs on embedded systems programmed using the Arduino IDE and custom firmware designed for real-time processing. Python is used extensively for developing data analytics algorithms, while OpenCV handles image processing tasks. Databases like MySQL or PostgreSQL store captured images, traffic data, and system logs, ensuring reliable data management.

SmartFlow integrates seamlessly with existing city-wide traffic management systems, ensuring compatibility and smooth operation. It also shares real-time data with emergency response systems and law enforcement databases to monitor traffic conditions, prioritize emergency vehicles, and enforce traffic regulations. This integration enhances emergency response efficiency and improves urban mobility.

### 4.1.6. User Documentation

The SmartFlow System is a technology-driven solution designed to optimize urban traffic control. It employs IoT devices such as cameras to monitor and analyze real-time traffic data, offering features like dynamic traffic signal control, predictive analytics, emergency response systems.

The Key Components are:

- Overview
- System Architecture
- User Interface
- Features

This documentation serves as a guide for users, offering detailed information on setup, features, troubleshooting, security measures, and ongoing maintenance of the SmartFlow system.

## 4.2.External Interface Requirements

### 4.2.1. Hardware Interfaces

The hardware infrastructure for a smart traffic management system using IoT comprises various components, including camera, traffic LEDs, a communication network, edge computing devices, Arduino mega, power supply systems.

- ESP32-camera: Camera captures the images of real time traffic and that is uploaded to database and that image is used to detect and count the number of vehicles in order to calculate the green signal delay and also to detect the presences of ambulance.
- Arduino Mega: Arduino is used for both programming the ESP32-camera and also to update the green signal calculated by algorithm on the bases of number of vehicles and also to change green signal when emergency vehicle is detected.
- Communication Network (Jumper wires): Establish a robust communication network to connect all the hardware components. This may include wired connections. Ensure secure and reliable communication for transmitting real-time data between sensors, controllers, and the central system.
- Power Supply: Ensure reliable power supply for all devices, considering backup options such as batteries or generators to prevent system failures during power outages.
- Traffic LED: Integrating traffic LEDs into a smart traffic management system can enhance the efficiency of traffic control and communication. Connect traffic LEDs to intelligent traffic signal controllers that are part of the overall smart traffic management system. Use real-time data from sensors and cameras to dynamically adjust signal timings and optimize traffic flow. Implement adaptive signal timings based on traffic conditions. Adjust the duration of green, yellow, and red lights dynamically to respond to changing traffic patterns.

### 4.2.2. Software Interfaces

The Arduino Integrated Development Environment (IDE) is an open-source software platform used for programming and developing applications for Arduino microcontroller boards. Arduino is a popular platform for hobbyists, students, and professionals to create a wide range of electronic projects and prototypes. Python IDLE used to write a code for YOLO algorithm to detect the vehicles, count the vehicles and calculate the green delay and also to detect the emergency vehicle.

I. Cross-Platform: Arduino IDE is available for Windows, macOS, and Linux, making it accessible to a broad user base.
II. Code Editor: It provides a text editor where you can write, edit, and save your Arduino sketches (code) written in C/C++.
III. Built-in Libraries: The IDE includes a collection of libraries that simplify interfacing with various hardware components, making it easier to control sensors, displays, motors, and other peripherals.
IV. Compiling and Uploading: Arduino IDE compiles your code into machine-readable binary files and uploads them to the Arduino board over a USB connection.
V. Serial Monitor: It has a built-in serial monitor that allows you to interact with your Arduino projects and view debugging information sent over the serial port.
VI. Board Manager: Arduino IDE supports various Arduino boards, and you can select the appropriate board from the "Boards Manager."
VII. Library Manager: You can easily add and manage third-party libraries for additional functionality.
VIII. Cross-Platform: Python IDLE is available for Windows, macOS, and Linux, making it accessible to a broad user base.
IX. Code Editor: It provides a text editor where you can write, edit, and save your Python scripts with syntax highlighting and auto-completion features to improve the coding experience.
X. Interactive Shell: IDLE includes an interactive Python shell that allows you to test and run Python code snippets in real-time, facilitating rapid prototyping and debugging.

### 4.2.3. Communications Interfaces

For the communication interface in your smart traffic management system using Arduino and various sensors/controllers, you can utilize a combination of wired and wireless communication protocols.

- **Wired Communication**

  Serial communication using UART (Universal Asynchronous Receiver-Transmitter) is a common and straightforward method to enable data exchange between different Arduino boards. UART allows for two-way communication between devices, using two wires: one for transmitting data (TX) and another for receiving data (RX).

- **Wireless Communication**

  The ESP32 offers robust wireless communication capabilities through both Wi-Fi and Bluetooth. In Wi-Fi Client Mode, the ESP32 connects to a network to send and receive data, ideal for IoT devices communicating with cloud servers. It can also function as a Wi-Fi Access Point, creating a local network for direct device connections without a router, and supports Wi-Fi Direct for direct device-to-device communication. For Bluetooth, the ESP32 includes Bluetooth Classic for continuous data streaming like audio, and Bluetooth Low Energy (BLE) for low-power, periodic data transfer, suitable for short-range communication with sensors and beacons.

## 4.3.System Features

In the SmartFlow system cameras placed at the traffic junction will capture the images of real time traffic that image is used by the algorithms to detect the number of vehicles to control the congestion and to detect the emergency vehicle to clear path for ambulance so that it will not experience the traffic and reach hospital as soon as possible.

### 4.3.1. Congestion Control

This feature focuses on managing and alleviating traffic congestion through real-time monitoring and adaptive traffic signal control.

**Functional Requirements**:

- **REQ-1: Real-Time Traffic Monitoring**:

  The system must continuously monitor traffic conditions using IoT sensors and cameras. Deploy ESP32-cameras at strategic points to capture traffic data and transmit it to the central server for detect and count the number of vehicles.

- **REQ-2: Dynamic Signal Adjustment**:

  The system must dynamically adjust traffic signal timings based on real-time traffic data within 10 seconds of detecting changes. The green delay calculated based on number of vehicles will be updated via Arduino Mega to optimize traffic flow and reduce congestion.

- **REQ-3: Data Analytics and Reporting**:

  The system must analyze traffic patterns and generate reports to identify congestion trends and inform future traffic management strategies. Utilize data analytics tools and techniques to process collected data and produce actionable insights.

### 4.3.2. Emergency Vehicle Detection

This feature ensures the swift and secure passage of emergency vehicles by detecting their presence and prioritizing their movement through intersections.

**Functional Requirements**:

- **REQ-1: Emergency Vehicle Identification**:

  The system must detect and identify emergency vehicles within 5 seconds of their approach. Use image processing and recognition algorithms on data captured by ESP32-cameras to identify emergency vehicles.

- **REQ-2: Signal Priority Activation**:

  When ambulance is detected, the system must activate signal priority to provide a clear path for emergency vehicles. This involves sending priority signals to the traffic control system via the Arduino Mega, which alters the signal phase to prioritize the emergency vehicle's route.

## 4.4.Other Nonfunctional Requirements

### 4.4.1. Performance Requirements

The SmartFlow traffic management system is designed with stringent performance requirements to ensure its effectiveness in real-time traffic monitoring and control.

The system must process real-time traffic data with a latency of less than 1 second. This requirement is essential for making swift decisions and adjustments to traffic signal timings, allowing the system to dynamically respond to changing traffic conditions. By meeting this requirement, SmartFlow optimizes traffic flow and minimizes congestion on roadways. A critical aspect of SmartFlow's functionality is the detection and prioritization of emergency vehicles within 5 seconds of their approach.

Swift identification and response are crucial for ensuring the rapid passage of emergency vehicles, thereby enhancing public safety and emergency management.

The system must dynamically adjust traffic signal timings within 10 seconds of detecting changes in traffic conditions. This requirement highlights the system's adaptive capabilities, enabling it to quickly respond to traffic fluctuations and optimize signal timings to reduce congestion. This rapid adaptation improves overall traffic flow efficiency, reducing delays for commuters and supporting the system's goal of providing a smooth and responsive smart traffic management experience.

### 4.4.2. Safety Requirements

The SmartFlow traffic management system must meet rigorous safety standards to mitigate potential risks of loss, damage, or harm resulting from its operation. A critical requirement is the prevention of traffic signal malfunctions, which could lead to accidents or traffic congestion. To address this, SmartFlow incorporates fail-safe mechanisms such as component redundancy, regular maintenance schedules, and real-time monitoring to promptly detect and rectify any malfunctions. These measures not only ensure continuous operation but also comply with local transportation authority regulations governing traffic signal control systems.

Another essential safety consideration is the reliable operation of emergency vehicle preemption to prevent accidents and delays. SmartFlow employs precise emergency vehicle detection algorithms, robust communication channels, and backup preemption systems to facilitate the swift and secure passage of emergency vehicles through intersections. Compliance with national and local emergency response standards ensures that the system effectively supports emergency management efforts without compromising public safety.

### 4.4.3. Security Requirements

The SmartFlow system is fortified with a set of security requirements aimed at safeguarding its integrity, data confidentiality, and overall functionality. Firstly, the system is mandated to implement robust access control mechanisms, ensuring that unauthorized access to sensitive components, data, and functionalities is strictly restricted. By enforcing stringent access controls, the system enhances its resilience against potential security threats, safeguarding critical elements from unauthorized manipulation or compromise. This measure not only contributes to the system's security posture but also aligns with industry best practices for access management.

The system places a strong emphasis on physical security. A specific requirement dictates the implementation of physical security measures, including access controls and surveillance, to protect critical hardware components and infrastructure. This multifaceted approach addresses not only virtual vulnerabilities but also the physical aspects of the system. Access controls and surveillance mechanisms serve as a deterrent to unauthorized physical access, bolstering the overall resilience of the smart traffic management system against both virtual and physical security threats.

# 4. System Design Description

## 5.1.Introduction

The purpose of this Software Design Document (SDD) is to provide a comprehensive overview of the design and architecture of the SmartFlow system. This document is intended for software developers, system architects, project managers, and stakeholders involved in the development and implementation of the SmartFlow system. It aims to ensure a clear understanding of the design considerations, system architecture, and the functionalities of the SmartFlow system.

The SmartFlow system is designed to manage urban traffic efficiently by prioritizing emergency vehicles and monitoring traffic rule violations. The system integrates IoT sensors, GPS technology, and real-time data analytics to achieve its objectives. It captures images using ESP32 cameras to detect congestion levels and identify emergency vehicles. The system then sends signals to an Arduino Mega to update traffic signals accordingly. Additionally, the system can detect traffic rule violations, capture images of offending vehicles, recognize license plates, and notify drivers of their infractions.

The SmartFlow system faces several constraints that must be addressed for optimal performance and compliance. Firstly, hardware limitations are significant, as the effectiveness of IoT sensors and cameras can be impacted by environmental conditions such as lighting and weather. Integration with existing traffic management infrastructure and emergency response systems is essential for seamless operation, requiring compatibility and possibly retrofitting current setups. Lastly, the system's design must consider scalability to manage increased data volumes and wider deployment areas, ensuring that performance remains consistent as the system expands.

## 5.2.System Overview

The SmartFlow system is designed to manage urban traffic efficiently by prioritizing emergency vehicles and monitoring traffic rule violations. This system addresses the increasing congestion in urban areas, which poses significant challenges for emergency response times and overall traffic management. The background of this

project lies in the necessity to enhance public safety and optimize traffic flow by utilizing modern technologies such as IoT and real-time data analytics.

### 5.2.1. System Architecture

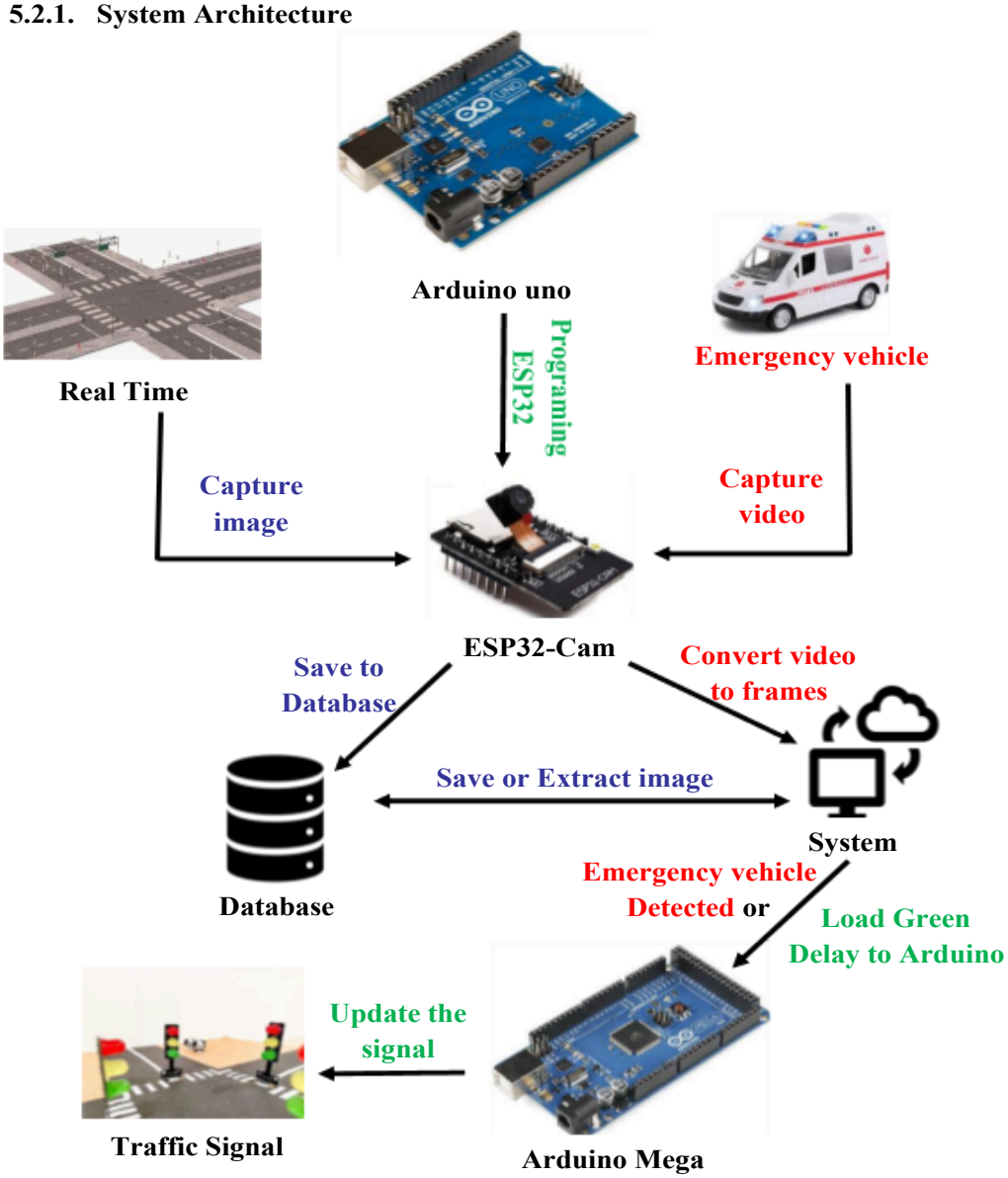



**Figure 1: Architecture diagram**

Above figure 1 represents the complete architecture of the SmartFlow system. The Arduino uno is used to program the ESP32 camera. Camera then captures the image of real traffic in one lane and it is stored in the database. Then that image is loaded to

the YOLO algorithm to count the number of vehicles in the image or a lane and based on the number of vehicles, green delay is calculated by using the formula

$$Green\_delay = (L + (vehicle\_count - 1) * (L + G)) / v$$

Where: L – the average length of the vehicle, vehicle_count – number of vehicles in one lane, G – inter-vehicle gap and v - Speed of vehicle m/s

### 5.2.2. Architectural Alternatives

Several architectural alternatives were considered for the SmartFlow system:

i. **Centralized Architecture:**
   - All data processing and decision-making occur at a single central server.
   - Pros: Simplifies data management and control.
   - Cons: Potential single point of failure and scalability issues.

ii. **Distributed Architecture:**
   - Data processing is distributed across multiple nodes closer to the data sources.
   - Pros: Enhanced scalability and fault tolerance.
   - Cons: Increased complexity in synchronization and data consistency.

iii. **Cloud-Based Architecture:**
   - Utilizes cloud services for data storage and processing.
   - Pros: Scalability and access to advanced cloud computing resources.
   - Cons: Dependency on internet connectivity and potential data privacy concerns.

### 5.2.3. Design Rationale

The selected architecture for SmartFlow is a hybrid approach combining centralized data processing with distributed sensors and local decision-making units. The rationale for this choice includes:

- Scalability: A hybrid architecture supports scalable deployment across various urban areas by distributing sensors and processing loads.
- Reliability: Combining centralized and distributed components ensures system robustness, reducing the risk of a single point of failure.

- Real-Time Processing: The architecture facilitates real-time data analysis and rapid response to traffic conditions and emergencies.

## 5.3.Functional Design

### 5.3.1. Describe the functionalities of the system

**Data flow Diagram:**

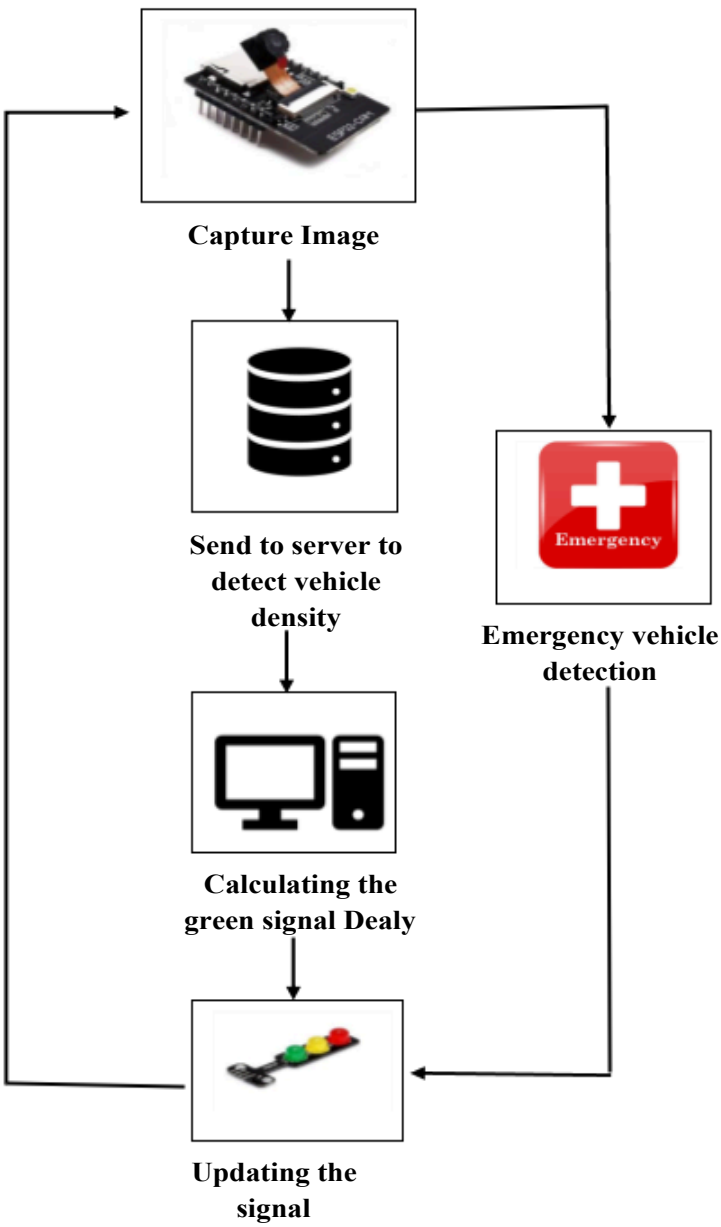



**Figure 2: Data flow diagram**

As shown in Figure 2, the ESP32 camera captures the image and stores it in the server, the YOLO algorithm processes the image detects the congestion, and calculates the green signal timing with the help of the Green_delay formula, according to that calculation the traffic signal is controlled. If there is any emergency vehicle is detected in the image taken by the camera then a green signal is turned on for the emergency vehicle.

**Use Case Diagram of SmartFlow:**

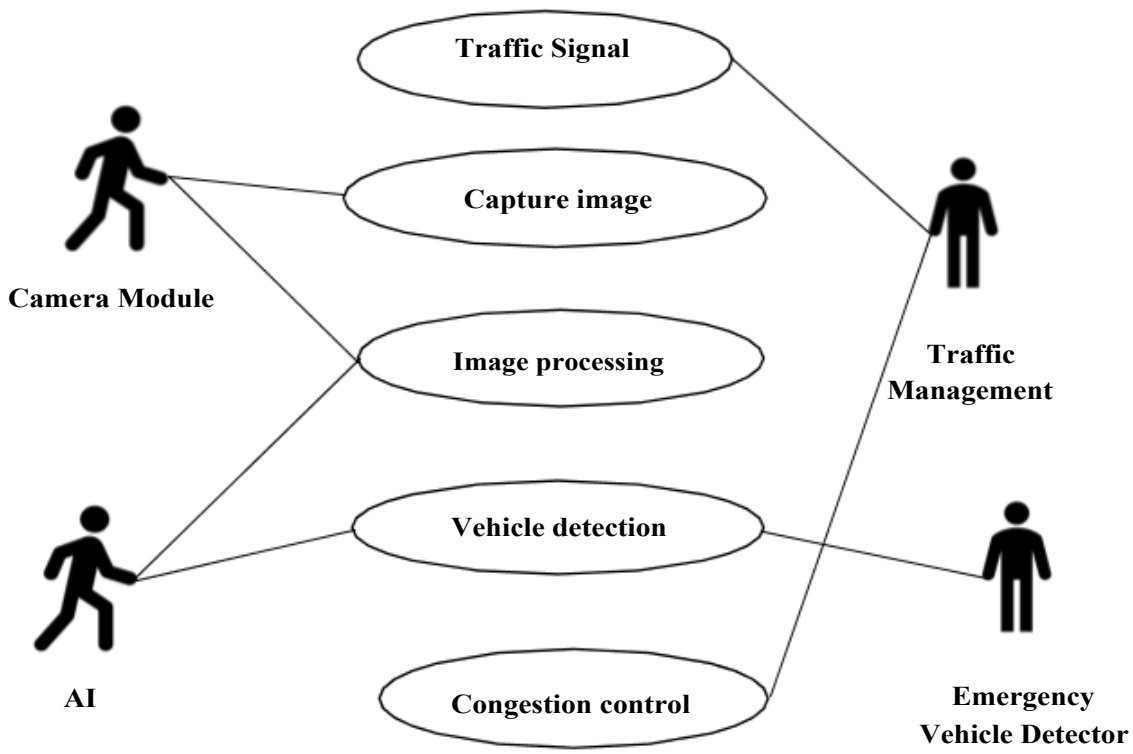



**Figure 3: Use case Diagram**

The figure 3 represents the use cases and actors in the SmartFlow system which can be described as follows

**Use Cases**

- **Traffic signal:** The traffic junction where the traffic lights are controlled by the traffic management system.
- **Capture image:** The cameras at junction takes images and send it to the system.
- **Image processing:** The AI algorithm processes the image to detect the number of vehicles and classify them.
- **Vehicle detection:** The AI algorithm uses data set and detect type and number of vehicles and detect the emergency vehicle.
- **Congestion control:** Traffic congestion, were there are more vehicles in one lane and a smaller number of vehicles in another lane leads to traffic.

**Actors**

- **Camera Module:** Captures the real time image or video to control the congestion based on the number of vehicles on a lane.
- **AI:** Artificial intelligence algorithm YOLO used to detect the vehicles and classify them to calculate the green delay.
- **Traffic Management:** The main system which controls the normal cycle of traffic and congestion control of the traffic, it uses the data given by the AI algorithm control the traffic by adjusting green delay.
- **Emergency Vehicle Detector:** The text detection algorithm is used to detect the emergency vehicle and inform system to turn on the green signal for emergency vehicle.

### 5.3.2. Behavioral design

**State Chart Diagram of SmartFlow:**

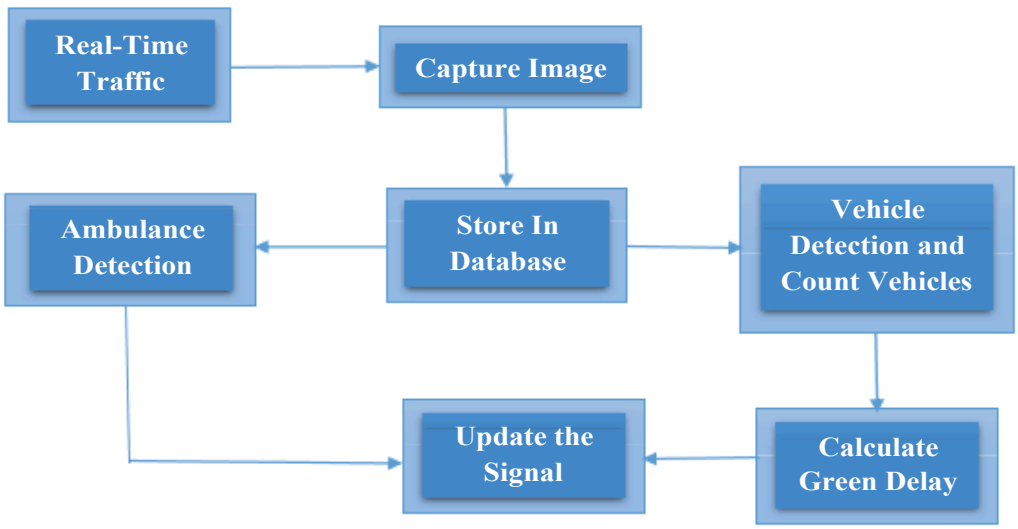



**Figure 4: State Chart Diagram**

As shown in the figure 4 the process starts with the system in an idle state, waiting to capture image in last 5 seconds of green delay. When the camera captures image of a lane, that image is stored in the database. The AI algorithm then retrieves these stored images to process them and count the number of vehicles in the lane. Based on the vehicle count, the system calculates the duration for the green signal in a lane. As the green signal period for a lane is about to end, the camera captures a new image of the next lane to prepare for the subsequent cycle.

Simultaneously, the AI algorithm continuously analyses the images to detect any emergency vehicles, such as ambulances. If an emergency vehicle is detected, the system immediately overrides the normal operation and activates the green signal for the lane with the emergency vehicle, allowing it to pass through without delay. After the emergency vehicle has passed, the system returns to its idle state, ready to capture the next set of images at the fixed interval, ensuring efficient traffic management and prioritization of emergency vehicles.

**Interaction diagram**

**Sequence Diagram of SmartFlow:**

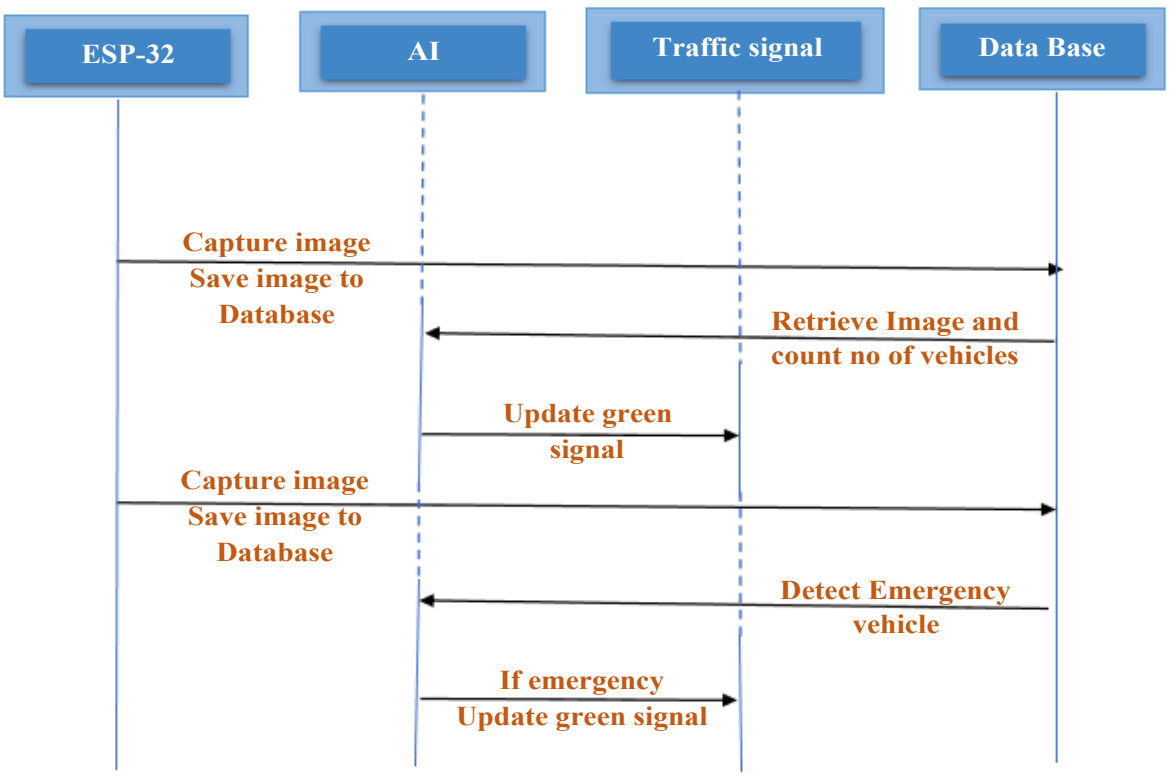



**Figure 5: Sequence Diagram**

As shown in the figure 5 camera captures the image and store it in the database, from database image is retrieved by AI algorithm to count the number of vehicles and calculate the green delay, this continues for all other lanes, when last few seconds of green delay camera takes picture of other lane. But the camera captures images of all lanes at fixed time of interval and that images are taken by the AI algorithm to detect the emergency vehicle this is continuous process so that emergency vehicles like ambulance will not suffer from traffic when the ambulance is detected the green signal is turned on.

# 5. Implementation

In the SmartFlow implementation, the system begins by establishing an RTSP connection to capture live video feeds for real-time traffic monitoring. It continuously retrieves frames from this stream, which are then stored in a database at regular intervals to facilitate historical traffic analysis. During operation, the system monitors the duration of green signals at intersections. If the delay in switching to green exceeds a set threshold, the system continues to observe traffic conditions until an appropriate time to change the signal timing is determined.

An essential feature of SmartFlow is its capability to detect emergency vehicles within the captured frames. Using specialized detection methods, the system identifies emergency vehicles promptly. Upon detection, it triggers actions such as adjusting signal timings and turn on the LED light to indicating ambulance has been detected. The process of SmartFlow is shown in the below algorithm steps. Step 7 n and step 9 are explained in the below modules.

**Algorithm for SmartFlow**

- **Step1:** Start
- **Step2:** Set up RSTP connection
- **Step3:** If failed to read frame from RSTP stream go to step 11 else go to step 5
- **Step4:** Initialize the data base connection
- **Step5:** While true repeat steps from 6 to
- **Step6:** Display live transmission from RSTP stream
- **Step7:** Capture the frames from the RSTP stream at regular interval of time and save to database
- **Step8:** If green delay is > 5 seconds go to step 7 else go to step 9
- **Step7:** Call function EmergencyVehicle()
- **Step8:** If emergency vehicle detected goto step12 else go to step 9
- **Step9:** Call function DelayCalculation(img)
- **Step10:** Update the green signal duration returned by DelayCalulation ()
- **Step11:** Turn on Yellow Led indicating failed to read from RSTP stream and go to step 14
- **Step12:** Turn on red led and update the green signal
- **Step13:** If keyboard interruption go to step 12 else go to step6
- **Step14:** Stop

## Module 1: Congestion control

Module one is for congestion control which is detecting and counting the number of vehicles to calculate the green delay in order to reduce the congestion. The process is as shown in the below algorithm.

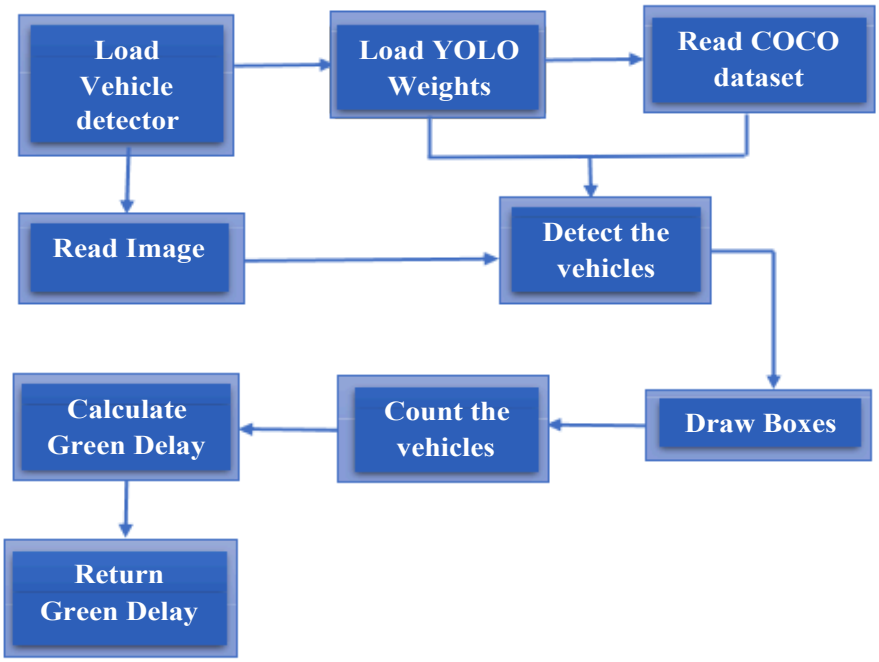



**Figure 6: Congestion control**

As shown in the figure 6 for congestion control, the Vehicle Detector Algorithm plays a pivotal role in managing traffic flow efficiently. This algorithm, integrated with YOLO weights and the COCO dataset, utilizes deep learning capabilities to detect vehicles within images extracted from the database. Employing OpenCV's cv2.dnn DetectionModel(), the algorithm performs vehicle detection by drawing bounding boxes around identified vehicles and counting these boxes to determine the total number of vehicles in the lane. This count forms the basis for calculating the green delay required for the traffic signal, considering factors such as average vehicle speed. The calculated green delay value is crucial for dynamically adjusting traffic signals via systems like the Arduino Mega, ensuring optimal traffic flow management in real-time scenarios.

**Algorithm for Congestion Control**

**DelayCalculation(img)**

- **Step1:** Load VehicleDetector()
- **Step2:** Initialize the source image and destination image path and speed
- **Step3:** Read the image
- **Step4:** Draw the boxes to vehicles
- **Step5:** count the number of vehicles
- **Step6:** Display the number of vehicles
- **Step7:** Save the image with bounding boxes
- **Step8:** Calculate the green delay

$$\boldsymbol{Green\_delay} = \frac{(\boldsymbol{L}+(\boldsymbol{vehicle_{count}}-\boldsymbol{1})*(\boldsymbol{L}+\boldsymbol{G})}{\boldsymbol{v}}$$

- **Step9:** Return speed

**VehicleDetector**

- **Step1:** Load YOLO4 model weights and configuration files using OpenCV
- **Step2:** Initialize the detection model using cv2.dn DetectionModel() and set input parameters, such as

  Size and scale
- **Step3:** Specify the classes that correspond to vehicles in the COCO dataset
- **Step4:** detect_vehicles(img)
  - **Step1:** Read the image
  - **Step2:** Detect the object in the image using YOLO4 model
  - **Step3:** Store the bounding boxes of the detected vehicles in a list
  - **Step4:** Return the list of boxes representing the detected vehicles.

In the module one, the process starts with loading the VehicleDetector module, which utilizes the YOLOv4 model for vehicle detection. This model is configured using OpenCV, allowing it to identify vehicles within a given image. Once an image is provided as input, the VehicleDetector module detects vehicles by applying the YOLOv4 model, which outputs bounding boxes around recognized vehicles. After detecting vehicles, the system proceeds to calculate the number of vehicles detected by simply counting the bounding boxes. This also calculate the green delay on the number of detected vehicles and average speed value.

## Module 2: Emergency Vehicle Detection

The module 2 is for emergency vehicle detection it uses EAST text detection module for detecting the text written on the ambulance. Once the ambulance is detected it will return true and the green signal for ambulance so that ambulance will reach hospital as soon as possible without any delay.

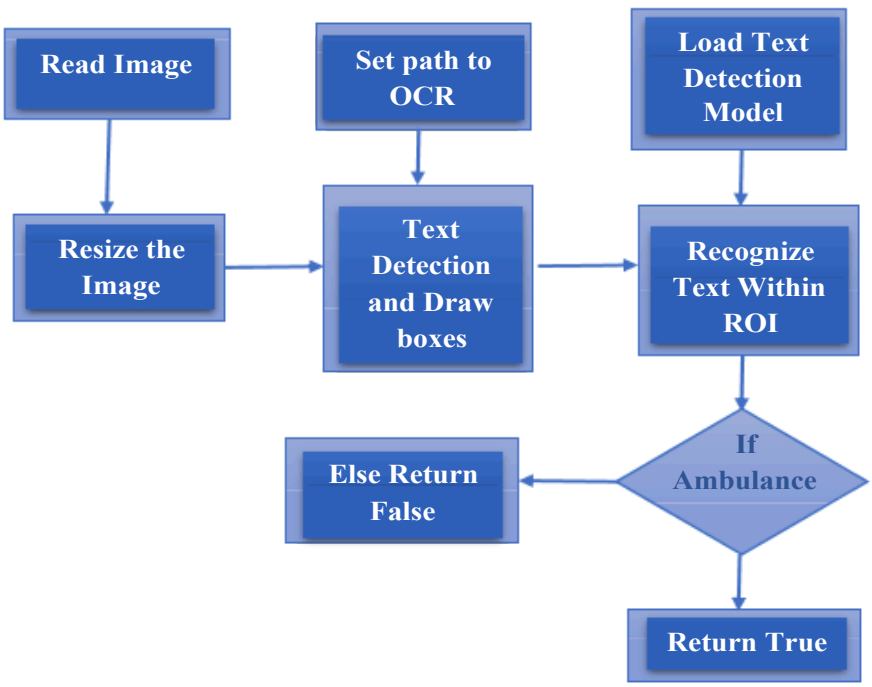



**Figure 7: Emergency Vehicle Detection**

As shown in the figure 7 emergency vehicle detection model utilizes a pre-trained text detection model known as frozen_east_text_detection.pb. When an image is fetched from the database, it is resized and then processed through the EAST model to detect text. Upon detection, bounding boxes are drawn around the identified text regions. The model extracts these regions of interest (ROI) from the original frame. The extracted text undergoes Optical Character Recognition (OCR) using Tesseract. The OCR results are stored in a variable to check that detected word is "AMBULANCE" OR NOT. If it is "AMBULANCE", the model returns true, indicating the presence of an emergency vehicle; otherwise, it returns false.

**Algorithm for Emergency Vehicle Detection**

**EmergencyVehicle**

- **Step1:** Set the path to the Tesseract OCR engine
- **Step2:** Load the pre-trained EAST text detection model (frozen_east_text_detection.pb) using OpenCV's DNN module.
- **Step3:** Read the image using from data base
- **Step4:** Resize the input frame to a fixed size to match the input size expected by the EAST model.
- **Step5:** Convert the resized frame to a blob suitable for input to the neural network (cv2.dnn.blobFromImage()).
- **Step6:** Pass the blob through the EAST model (net.forward()) to obtain predictions draw boxes.
- **Step7:** Text Recognition (OCR) Extract the region of interest (ROI) from the original frame. Apply Tesseract OCR (pytesseract.image_to_string()) to recognize text within the ROI.
- **Step8:** Return true if text is 'AMBULANCE' else false

Module two works on the text detection and recognition system, the process begins by setting up the Tesseract OCR engine path, which allows the system to leverage Tesseract for Optical Character Recognition (OCR). The pre-trained EAST text detection model, specifically the frozen_east_text_detection.pb file, is then loaded using OpenCV's Deep Neural Network module.

The system reads the input image from a database, which serves as the source of data for processing. The input frame is resized to a fixed dimension that matches the model's expected input size. Once resized, the frame is converted into a blob using the cv2.dnn.blobFromImage() function.

The blob is then passed through the EAST model using the net.forward() function to obtain text detection and bounding boxes drawn around the detected text areas in the image. And the system extracts the regions of interest (ROIs) from the original frame based on the bounding boxes identified by the EAST model. These ROIs are then processed using Tesseract OCR through the pytesseract.image_to_string() function, which recognizes and converts the text within these regions into a readable string format.

# 6. Testing

## 7.1.Description of Testing

### Unit Testing

Each functionality of the SmartFlow is tested first the camera is connected to check whether video or image is captured and sending to the system. Next the YOLO algorithm is tested using the test images to detect and count the number of vehicles. The normal working of the traffic signals is tested. The Emergency vehicle detection using text detection module is tested.

### Integration

The emergency vehicle detection module and traffic signals both are combined and tested to check the signals turning green immediately for emergency vehicle to pass through junction and camera module and algorithm are combined and tested to solve the traffic congestion based on the number of vehicles present in a lane.

### System testing

End to end functionality of the system is tested by integrating congestion module and emergency vehicle detection module so that traffic congestion problem is solved based on the number of vehicles present on the lane and also the emergency vehicles like ambulance is given high priority at traffic junction so that it reaches the hospital as soon as possible.

## 7.2.Test Cases

**Table 1: Test Cases for Individual Modules**

| Test case # | Test case Name | Test case Description | Inputs | Expected Output | Actual Output | Status |
|---|---|---|---|---|---|---|
| 1 | Testing ESP32 Camera | To check the live streaming and capturing the images and storing in the database | Connecting camera to Arduino uno and uploading code | RSTP live streaming URL | RSTP live streaming URL | Pass |
| 2 | Normal Traffic flow | To check uploading code to Arduino through PyCharm works or not | Connecting traffic LEDS | The duration of green signal at each lane is same | Error | Fail |
| 3 | Executing YOLO algorithm | To check YOLO algorithm, detect and count the number of vehicles or not | Some images of vehicles in traffic | Draw boxes for detected vehicles and display count | Output image with boxes drawn for detected vehicles | Pass |
| 4 | Text detection | Testing the EAST module for ambulance test detection from image | Image of ambulance | Draw boxes around detected text ambulance | Error | Fail |

**Table 2: Test cases for Combined Modules**

| | | | | | | |
|---|---|---|---|---|---|---|
| 5 | Congestion control | Testing code to count number of vehicles form image taken from ESP32 camera and change green signal | Connect camera and database | Green delay on the bases of number of vehicles | Green delay on the bases of number of vehicles | Pass |
| 6 | Ambulance Detection | Testing code to detect the ambulance from image taken by camera and change green signal | Connect camera and database | Turn on LED to indicate presence of ambulance and change green signal | Error | Fail |
| 7 | Emergency vehicles detection | Combining all the models to test for ambulance detection | Connect camera and database | Turn on the LED to indicate the presence of an ambulance and change the green signal | Error | Fail |
| 8 | SmartFlow: Congestion control | Combining all the models to test for congestion control | Connect camera and database | Green delay on the basis of number of vehicles | Green delay on the basis of number of vehicles | Pass |

# 7. Reports

The SmartFlow system successfully established an RTSP connection for continuous live video feed to monitor real-time traffic. It efficiently captured and stored frames in a database at regular intervals, enabling robust historical traffic analysis. The system demonstrated reliable performance in monitoring and adjusting signal timings at intersections, effectively reducing delays. A key feature of SmartFlow is its ability to detect emergency vehicles within the captured frames. This was achieved with high accuracy, promptly triggering actions such as adjusting signal timings and activating LED lights to indicate the presence of an emergency vehicle. The historical data storage allowed for insightful analysis of traffic patterns, revealing peak hours and common bottlenecks. Despite some challenges like occasional technical issues and environmental factors affecting performance, the system proved to be a valuable tool for improving traffic management and safety. Overall, SmartFlow demonstrated significant potential in optimizing traffic flow and responding effectively to emergencies.

To analyze the performance and impact of the SmartFlow system, we have visualized key data points using graphs. These graphs illustrate the system's efficiency in real-time traffic monitoring, signal timing adjustments, and emergency vehicle detection.

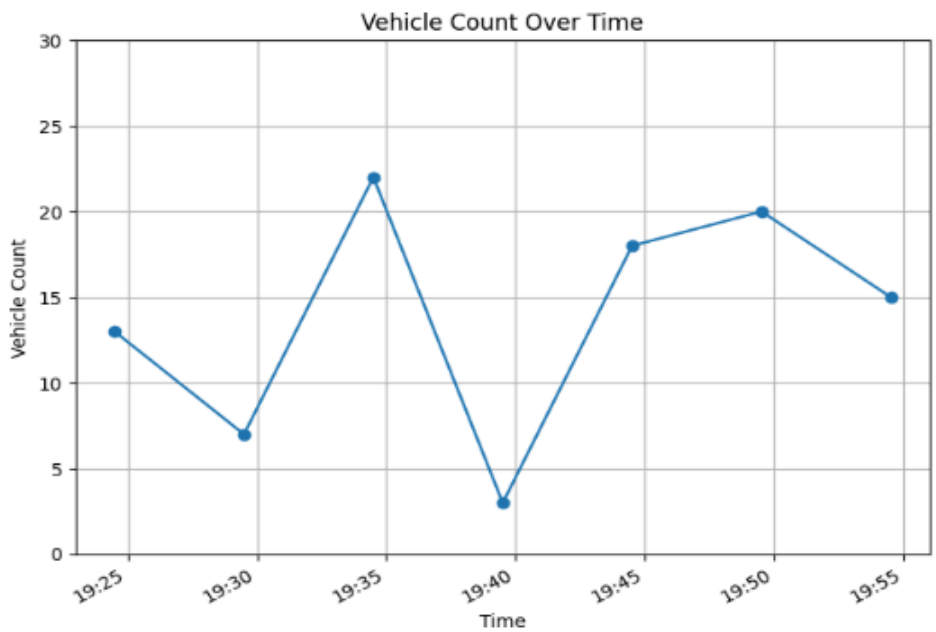



**Figure 8: Vehicle Count over Time**

Figure 8 illustrates the vehicle count over time across four lanes, each experiencing different delays for red, green, and yellow signals. The graph reveals significant traffic volume trends, highlighting peak periods where vehicle counts are highest. Lane 1 consistently shows the lowest vehicle count, correlating with its minimal signal delays, indicating efficient traffic flow. In contrast, Lane 4 displays the highest vehicle count and substantial delays, particularly for red and green signals, suggesting a need for SmartFlow to optimize signal timings to better manage heavy traffic.

Lanes 2 and 3, despite having green signals, exhibit significant delays, indicating inefficiencies in vehicle movement even when signals are green. This results in higher vehicle counts over extended periods. SmartFlow can address these inefficiencies by adjusting green signal durations in real time to ensure smoother traffic flow. The pronounced red delays in Lanes 3 and 4 contribute to prolonged wait times, leading to increased vehicle counts during these phases. By reducing red delays through adaptive signal control, SmartFlow can alleviate congestion and enhance traffic throughput.


**Figure 9: Traffic Analysis with Fixed Green Delay**

Figure 9 represents the green, yellow, and red delays in normal traffic junctions when the green delay is fixed for all 4 lanes which is 20 seconds. The waiting time that is red delay will be more.


**Figure 10 : Dynamic Green Delay Traffic Analyses**

Figure 10 represents the green, yellow, and red delay for all 4 lanes when traffic congestion is controlled by dynamic green delay that is based on the number of vehicles present in each lane. The green delay is calculated by using the formula

$$Green\_delay = \frac{(L+(vehicle_{count}-1)*(L+G)}{v}$$

where v is the vehicle speed (10 m/s), is the average length of vehicles (5 meters), and G is the inter-vehicle gap (2 meters). The vehicle counts for lanes 1 to 4 are 3, 15, 20, and 18, respectively. Using this formula, the green delays for the lanes are calculated as 1.9, 14.5, 13.8, and 12.4 seconds, respectively. The total green delay across all lanes sums up to 42.6 seconds. Adding a yellow delay of 2 seconds per lane results in a total time of 50.6 seconds.

In contrast, if a fixed green delay of 20 seconds per lane is used, the total green delay for all lanes is 60 seconds. Including the yellow delay of 8 seconds across all lanes brings the total time to 68 seconds. This comparison shows that the waiting time, represented by the red delay, is significantly higher with fixed green delays. For instance, in the dynamic green delay scenario, the red delay for Lane 2 is 3.9 seconds, while it would be 22 seconds with a fixed green delay. Similarly, for Lane 3, the red delay is 20.4 seconds with a dynamic green delay compared to 44 seconds with a fixed green delay.

These results indicate that the SmartFlow system, which employs dynamic green delays, effectively reduces waiting times, manages traffic congestion more efficiently, and also conserves energy. By adjusting the green signal duration based on real-time traffic conditions, SmartFlow optimizes traffic flow, reduces unnecessary idling, and enhances overall intersection performance.

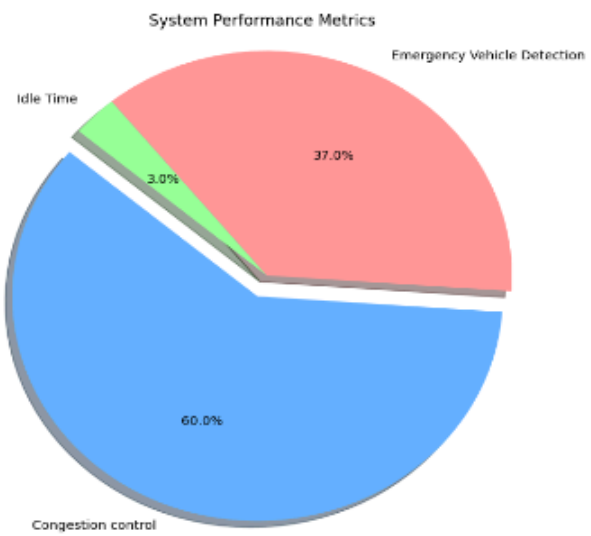



**Figure 11: System Performance Metrics**

Figure 11 represents the system performance metrics. The Blue color indicates the congestion control that is capturing the image that image is used by the YOLO algorithm for detecting and counting the number of vehicles and calculating the green delay. The red color represents the emergency vehicle detection by the EAST text detection module from the image captured by the camera to provide a smooth flow for emergency vehicles like ambulances. The percentage of emergency vehicle detection depends on the arrival of ambulances in the traffic junctions. Most of the time system does the congestion control. The system will be ideal only for few minutes so it shown only three percentage in the graph.

# 8. Conclusion

The proposed system SmartFlow will control the traffic congestion based on the density that is by calculating the number of vehicles in the lane by taking the image and calculate green delay to adjust green delay according to the number of vehicles and the emergency vehicle is detected from image taken from camera and green signal is given to ambulance so that it does not experience any traffic or delay in reaching hospital.

The project SmartFlow provides an effective solution for rapid growth of traffic flow particularly in big cities which is increasing day by day and traditional systems have some limitations as they fail to manage current traffic effectively. The SmartFlow is proposed to control road traffic situations more efficiently and effectively. It changes the signal timing intelligently according to traffic density on the particular roadside and regulates traffic flow by capturing the image. This project also makes sure that emergency vehicle like ambulance will not suffer from any interruption due to traffic signal so that ambulance can reach hospital as soon as possible.

# 9. Future Scope and Further Enhancement of The Project

The proposed system aims to control traffic congestion using cameras to detect emergency vehicles. Building upon this system, we can implement additional features to enhance its functionality and effectiveness. One significant enhancement is accident detection. By integrating sensors and advanced image processing techniques, the system can detect accidents in real-time. When an accident is detected, the system can automatically inform the nearest ambulance service, ensuring a prompt emergency response.

Furthermore, we can implement Li-Fi (Light Fidelity) navigation to assist ambulance drivers. Li-Fi technology uses light signals for communication, providing a reliable and fast method to transmit data. This can be used to guide ambulance drivers to the nearest hospital by providing the optimal route with the least traffic congestion. The Li-Fi navigation system can dynamically update the route based on real-time traffic data, ensuring that ambulances reach their destinations as quickly as possible.

Following traffic rules is crucial for maintaining order and safety on the roads. The system can be enhanced to detect traffic rule violations effectively. By utilizing high-resolution cameras, the system can capture images of vehicles that violate traffic rules, such as running red lights, speeding, or making illegal turns. These images can then be processed using license plate recognition technology to identify the offending vehicles. Once the vehicle is identified, the system can automatically send a message to the driver, informing them of the violation and any applicable penalties.

Incorporating these features will significantly improve the proposed system's ability to manage traffic, respond to emergencies, and enforce traffic rules.

# 10. User Manual

## Introduction

SmartFlow is an advanced traffic management system designed for controlling the real-time traffic. It captures live video and images, analyzes traffic conditions, and dynamically adjusts signal timings to optimize traffic flow. The system also includes a critical feature to detect emergency vehicles and prioritize their movement.

## Features

- **Real-time Traffic Monitoring:** Establishes an RTSP connection to capture live video feeds for continuous traffic monitoring.
- **Historical Traffic Analysis:** Stores frames at regular intervals in a database to facilitate the analysis of past traffic patterns.
- **Dynamic Signal Adjustment:** Monitors green signal duration and adjusts timing based on current traffic conditions.
- **Emergency Vehicle Detection:** Identifies emergency vehicles in the captured frames and adjusts signals to prioritize their passage.

## System Operation

1. **Initialize RTSP Connection:** The system begins by setting up an RTSP connection to capture live video feeds.
2. **Frame Retrieval:** Continuously retrieves frames from the live video stream.
3. **Frame Storage:** Stores frames in a database at regular intervals for historical traffic analysis.
4. **Monitor Signal Duration:** Observes the duration of green signals at intersections.
5. **Signal Timing Adjustment:** Determines the appropriate time to change the signal timing based on observed traffic conditions.
6. **Emergency Vehicle Detection:** Utilizes specialized detection methods to identify emergency vehicles within the captured frames.
7. **Trigger Actions:** Upon detecting an emergency vehicle, the system adjusts signal timings and activates LED lights to indicate the presence of an emergency vehicle.

**Emergency Vehicle Detection**

1. **Detection Methods:** Uses text detection EAST module for detecting the text ambulance written on the emergency vehicle.
2. **Response Actions:** Adjusts signal timings to prioritize the emergency vehicle and activates LED lights as an indicator.

**Congestion Control**

1. **Traffic Analysis:** Uses YOLO algorithm to detect and count the number of vehicles and calculate the green delay.
2. **Adaptive Control:** Ensures smooth traffic flow by dynamically adjusting signal timings based on number of vehicles.