# Introduction

## Overview

* 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 for being incapable of handling extensive traffic congestion and eventually leads to the environmental pollution for an extended period of vehicle emission. Adequate pre-measures and proper planning can help to reduce the number of traffic problems and manage an increased number of vehicles on the road. Traffic system utilize the concept of automation with IoT is called as “Smart Traffic”. Smart Traffic 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. This system relies on various sensors placed strategically throughout the road network to monitor traffic conditions
* This system will monitor the traffic using camera.
* The system can control traffic signals at intersections dynamically based on real-time traffic data.
* Adaptive traffic signal systems adjust signal timings to minimize waiting times and reduce idling
* Reducing congestion and energy consumption at intersection.
* Ensuring immediate clearance for emergency vehicles. Facilitating safer and shorter commute time.
* The emergency vehicle is detected, which gives ambulances priority to pass through traffic lights.

## Problem Definition

In contemporary times, the escalating demand for transportation coupled with rapid urbanization places an immense strain on existing infrastructure, particularly in managing road transportation. One of the most pressing challenges stemming from this is the widespread occurrence of traffic jams, predominantly concentrated in urban areas. These jams give rise to a multitude of issues, including heightened levels of noise and air pollution, as well as significant delays in travel time. The current state of congestion poses a substantial threat to various facets of life, including the economy, environment, and overall well-being. A significant contributor to traffic congestion is the malfunctioning of traffic lights and other related infrastructural deficiencies. Such inadequacies result in prolonged red-light delays, exacerbating congestion and its associated negative impacts. This congestion not only squanders valuable productive time but also leads to the wastage of fossil fuels, exacerbates pollution levels, and inflicts substantial economic losses.

The existing traffic signal systems deployed across cities, with their fixed predetermined timings for red and green signals, prove insufficient in addressing the aforementioned challenges. Recognizing this limitation, numerous endeavors have been made to imbue traffic lights with intelligence, enabling them to respond dynamically to the density of vehicles on the road.

The proposed system will solve this problem. The camera’s installed in the traffic signals will send AI to count number of vehicles so that based on number of vehicles the traffic lights delay is adjusted.

# Hardware and Software Requirements

## Hardware Requirements

* Arduino MEGA
* ESP32-cam
* Ultrasonic Sensors
* RFID Module
* Traffic LEDs
* Jumper wires

## Software Requirements

* Arduino IDE
* Efp-idf
* Google Cloud vision

# Software Requirements Specification

## System Features

The cameras installed at the traffic junctions will capture videos and photos and send it to the system. Using AI algorithm count the number of vehicles and adjust the traffic signal. Using RFID tags emergency vehicle is detected.

### Traffic Analysis

**REQ-1: Real-time Video Feed Capture**

The system will capture the video or image of the lane for which the signal is going to be green in next 5 seconds.

**REQ-2: Sending Image to system**

The image taken by the camera is sent to AI algorithm in order to calculate the number of vehicles.

### Object Detection and Classification

**REQ-1: Detecting vehicles**

Algorithm is used to detect the number of vehicles in the image and classifying the vehicles such as car, bus, bike, truck etc.

**REQ-2: Traffic Flow Analysis**

Analyzing the traffic flow Based on the number and type of vehicle calculate the green delay of the signal for one lane in order to avoid the congestion.

### Emergency Vehicle Detection

**REQ-1: Detecting the Emergency vehicle**

Using RFID tags the emergency vehicle like ambulance is detected.

**REQ-2: Traffic Flow Analysis**

As soon as ambulance is detected the signal for the particular lane is turned to green so that the ambulance reaches the hospital without facing any traffic.

# System Design Description (SDD)

## System Overview

**System Architecture:**

**Smart Flow**

**Emergency vehicle Detection**

**Capturing the image**

**Normal cycle for Traffic signal**

**Congestion Control**

The Smartflow system consist of normal cycle for traffic signal, congestion control using camera, detecting the number of vehicles and classifying them using AI YOLO algorithm and emergency vehicle detection using RFID tag. For camera the code is loaded with the help of Ardino board once and then the images taken by the camera are viewed in the AI screen this image is sent to algorithm as input and algorithm gives the vehicle count and also the value of green delay. Then the signals are managed. The RFID tag is used to detects the emergency vehicle and inform the system so that green signal is turned for emergency vehicle.

## Data Set Description

The number of vehicles detection and the classification vehicles is processed by YOLO algorithm. The data set consist of all different categories of vehicles is given as input which has lists of vehicles such as car, bus, truck, bike etc. The image taken by the camera is stored on the system.

## Functional Design

### Describe the functionalities of the system:

**Use case diagram**

**Capture image**

**Traffic Signal**

**Vehicle detection**

**Congestion control**

**Image processing**

**Camera Module**

**Traffic**

**Management**

**AI**

**Emergency Vehicle detector**

**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. The RFID is used to detect the emergency vehicle.

**Congestion control:** Traffic congestion , were there are more vehicles in one lane and less 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 and RFID tag to control the traffic by adjusting green delay.

**Emergency Vehicle Detector:** The RFID tag is used to detect the emergency vehicle and inform system to turn on the green signal for emergency vehicle

### Behavioral design:

**Emergency vehicle detection**

**Updating the signal**

**Capture Image**

**Sent to server to detect vehicle density**

**Calculating the green signal**

### The camera captures the image and store it in the server, then YOLO algorithm processes the image and detect the congestion and calculate the green signal timing, according that calculation the traffic signal is controlled. If there is any emergency vehicle is arriving towards the traffic junction then RFID tag detects the vehicle and green signal is turned on for the emergency vehicle.

# Implementation

**Setup.py**

from Cython.Build import cythonize

import numpy

import os

import imp

VERSION = imp.load\_source('version', os.path.join('.', 'darkflow', 'version.py'))

VERSION = VERSION.\_\_version\_\_

if os.name =='nt' :

    ext\_modules=[

        Extension("darkflow.cython\_utils.nms",

            sources=["darkflow/cython\_utils/nms.pyx"],

            #libraries=["m"] # Unix-like specific

            include\_dirs=[numpy.get\_include()]

        ),

        Extension("darkflow.cython\_utils.cy\_yolo2\_findboxes",

            sources=["darkflow/cython\_utils/cy\_yolo2\_findboxes.pyx"],

            #libraries=["m"] # Unix-like specific

            include\_dirs=[numpy.get\_include()]

        ),

        Extension("darkflow.cython\_utils.cy\_yolo\_findboxes",

            sources=["darkflow/cython\_utils/cy\_yolo\_findboxes.pyx"],

            #libraries=["m"] # Unix-like specific

            include\_dirs=[numpy.get\_include()]

        )

    ]

elif os.name =='posix' :

    ext\_modules=[

        Extension("darkflow.cython\_utils.nms",

            sources=["darkflow/cython\_utils/nms.pyx"],

            libraries=["m"], # Unix-like specific

            include\_dirs=[numpy.get\_include()]

        ),

        Extension("darkflow.cython\_utils.cy\_yolo2\_findboxes",

            sources=["darkflow/cython\_utils/cy\_yolo2\_findboxes.pyx"],

            libraries=["m"], # Unix-like specific

            include\_dirs=[numpy.get\_include()]

        ),

        Extension("darkflow.cython\_utils.cy\_yolo\_findboxes",

            sources=["darkflow/cython\_utils/cy\_yolo\_findboxes.pyx"],

            libraries=["m"], # Unix-like specific

            include\_dirs=[numpy.get\_include()]

        )

    ]

else :

    ext\_modules=[

        Extension("darkflow.cython\_utils.nms",

            sources=["darkflow/cython\_utils/nms.pyx"],

            libraries=["m"] # Unix-like specific

        ),

        Extension("darkflow.cython\_utils.cy\_yolo2\_findboxes",

            sources=["darkflow/cython\_utils/cy\_yolo2\_findboxes.pyx"],

            libraries=["m"] # Unix-like specific

        ),

        Extension("darkflow.cython\_utils.cy\_yolo\_findboxes",

            sources=["darkflow/cython\_utils/cy\_yolo\_findboxes.pyx"],

            libraries=["m"] # Unix-like specific

        )

    ]

setup(

    version=VERSION,

    name='darkflow',

    description='Darkflow',

    license='GPLv3',

    url='https://github.com/thtrieu/darkflow',

    packages = find\_packages(),

    scripts = ['flow'],

    ext\_modules = cythonize(ext\_modules)

)

**Vehicle\_detection.py**

import cv2

from darkflow.net.build import  TFNet

import matplotlib.pyplot as plt

import os

options={

   'model':'./cfg/yolo.cfg',        #specifying the path of model

   'load':'./bin/yolov2.weights',   #weights

   'threshold':0.3                  #minimum confidence factor to create a box, greater than 0.3 good

}

tfnet=TFNet(options)

inputPath = os.getcwd() + "/test\_images/"

outputPath = os.getcwd() + "/output\_images/"

def detectVehicles(filename):

   global tfnet, inputPath, outputPath

   img=cv2.imread(inputPath+filename,cv2.IMREAD\_COLOR)

   # img=cv2.cvtColor(img,cv2.COLOR\_BGR2RGB)

   result=tfnet.return\_predict(img)

   # print(result)

   for vehicle in result:

      label=vehicle['label']   #extracting label

      if(label=="car" or label=="bus" or label=="bike" or label=="truck" or label=="rickshaw"):    # drawing box and writing label

         top\_left=(vehicle['topleft']['x'],vehicle['topleft']['y'])

         bottom\_right=(vehicle['bottomright']['x'],vehicle['bottomright']['y'])

         img=cv2.rectangle(img,top\_left,bottom\_right,(0,255,0),3)    #green box of width 5

         img=cv2.putText(img,label,top\_left,cv2.FONT\_HERSHEY\_COMPLEX,0.5,(0,0,0),1)   #image, label, position, font, font scale, colour: black, line width

   outputFilename = outputPath + "output\_" +filename

   cv2.imwrite(outputFilename,img)

   print('Output image stored at:', outputFilename)

   # plt.imshow(img)

   # plt.show()

   # return result

for filename in os.listdir(inputPath):

   if(filename.endswith(".png") or filename.endswith(".jpg") or filename.endswith(".jpeg")):

      detectVehicles(filename)

print("Done!")

**Simulation.py**

def generateVehicles():

    while(True):

        vehicle\_type = random.randint(0,4)

        if(vehicle\_type==4):

            lane\_number = 0

        else:

            lane\_number = random.randint(0,1) + 1

        will\_turn = 0

        if(lane\_number==2):

            temp = random.randint(0,4)

            if(temp<=2):

                will\_turn = 1

            elif(temp>2):

                will\_turn = 0

        temp = random.randint(0,999)

        direction\_number = 0

        a = [400,800,900,1000]

        if(temp<a[0]):

            direction\_number = 0

        elif(temp<a[1]):

            direction\_number = 1

        elif(temp<a[2]):

            direction\_number = 2

        elif(temp<a[3]):

            direction\_number = 3

        Vehicle(lane\_number, vehicleTypes[vehicle\_type], direction\_number, directionNumbers[direction\_number], will\_turn)

        time.sleep(0.75)

def simulationTime():

    global timeElapsed, simTime

    while(True):

        timeElapsed += 1

        time.sleep(1)

        if(timeElapsed==simTime):

            totalVehicles = 0

            print('Lane-wise Vehicle Counts')

            for i in range(noOfSignals):

                print('Lane',i+1,':',vehicles[directionNumbers[i]]['crossed'])

                totalVehicles += vehicles[directionNumbers[i]]['crossed']

            print('Total vehicles passed: ',totalVehicles)

            print('Total time passed: ',timeElapsed)

            print('No. of vehicles passed per unit time: ',(float(totalVehicles)/float(timeElapsed)))

            os.\_exit(1)

class Main:

    thread4 = threading.Thread(name="simulationTime",target=simulationTime, args=())

    thread4.daemon = True

    thread4.start()

    thread2 = threading.Thread(name="initialization",target=initialize, args=())    # initialization

    thread2.daemon = True

    thread2.start()

   # Colours

    black = (0, 0, 0)

    white = (255, 255, 255)

    # Screensize

    screenWidth = 1400

    screenHeight = 800

    screenSize = (screenWidth, screenHeight)

    # Setting background image i.e. image of intersection

    background = pygame.image.load(r'C:/Users/ASUS/Desktop/Miniproject/Code/YOLO/darkflow/images/mod\_int.png')

    screen = pygame.display.set\_mode(screenSize)

    pygame.display.set\_caption("SIMULATION")

    # Loading signal images and font

    redSignal = pygame.image.load(r'C:/Users/ASUS/Desktop/Miniproject/Code/YOLO/darkflow/images/signals/red.png')

    yellowSignal = pygame.image.load(r'C:/Users/ASUS/Desktop/Miniproject/Code/YOLO/darkflow/images/signals/yellow.png')

    greenSignal = pygame.image.load(r'C:/Users/ASUS/Desktop/Miniproject/Code/YOLO/darkflow/images/signals/green.png')

    font = pygame.font.Font(None, 30)

    thread3 = threading.Thread(name="generateVehicles",target=generateVehicles, args=())    # Generating vehicles

    thread3.daemon = True

    thread3.start()

    while True:

        for event in pygame.event.get():

            if event.type == pygame.QUIT:

                sys.exit()

        screen.blit(background,(0,0))

        for i in range(0,noOfSignals):  # display signal and set timer according to current status: green, yello, or red

            if(i==currentGreen):

                if(currentYellow==1):

                    if(signals[i].yellow==0):

                        signals[i].signalText = "STOP"

                    else:

                        signals[i].signalText = signals[i].yellow

                    screen.blit(yellowSignal, signalCoods[i])

                else:

                    if(signals[i].green==0):

                        signals[i].signalText = "SLOW"

                    else:

                        signals[i].signalText = signals[i].green

                    screen.blit(greenSignal, signalCoods[i])

            else:

                if(signals[i].red<=10):

                    if(signals[i].red==0):

                        signals[i].signalText = "GO"

                    else:

                        signals[i].signalText = signals[i].red

                else:

                    signals[i].signalText = "---"

                screen.blit(redSignal, signalCoods[i])

        signalTexts = ["","","",""]

        # display signal timer and vehicle count

        for i in range(0,noOfSignals):

            signalTexts[i] = font.render(str(signals[i].signalText), True, white, black)

            screen.blit(signalTexts[i],signalTimerCoods[i])

            displayText = vehicles[directionNumbers[i]]['crossed']

            vehicleCountTexts[i] = font.render(str(displayText), True, black, white)

            screen.blit(vehicleCountTexts[i],vehicleCountCoods[i])

        timeElapsedText = font.render(("Time Elapsed: "+str(timeElapsed)), True, black, white)

        screen.blit(timeElapsedText,(1100,50))

        # display the vehicles

        for vehicle in simulation:

            screen.blit(vehicle.currentImage, [vehicle.x, vehicle.y])

            # vehicle.render(screen)

            vehicle.move()

        pygame.display.update()

Main()

# Testing

## Description of Testing

**Unit Testing**

Each functionality of the SmartFlow is tested first the camera connectivity is connected to check video or image processing and sending to the system. Next the algorithm is tested using the test images to detect the number of vehicles and classify them. The normal working of the traffic signals is tested. The RFID tags are tested to detect the emergency vehicle.

**Integration**

The RFID tag 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.

**System testing**

End to end functionality of the system is tested by integrating camera module and RFID with algorithm 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.

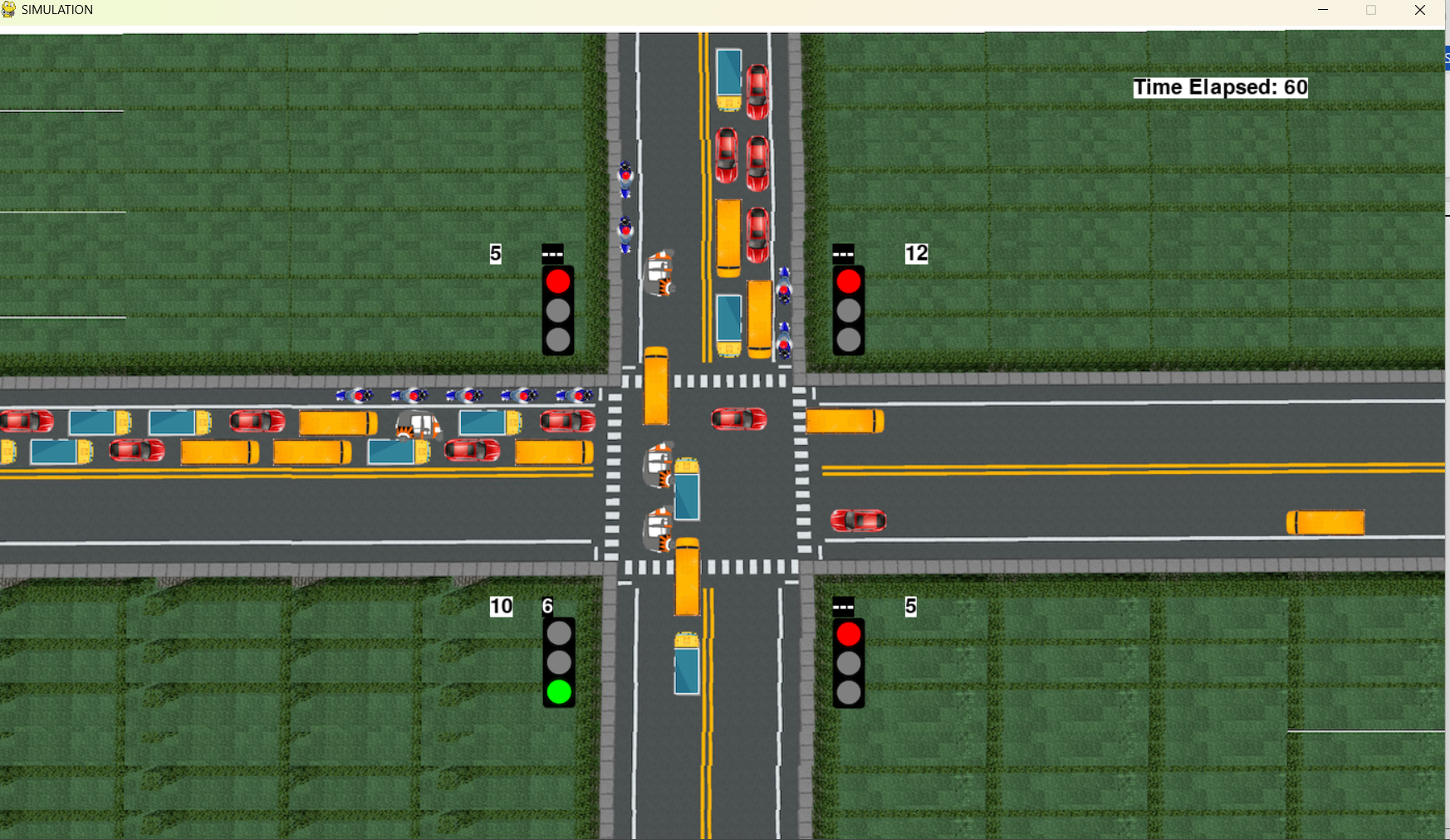
## Test Cases

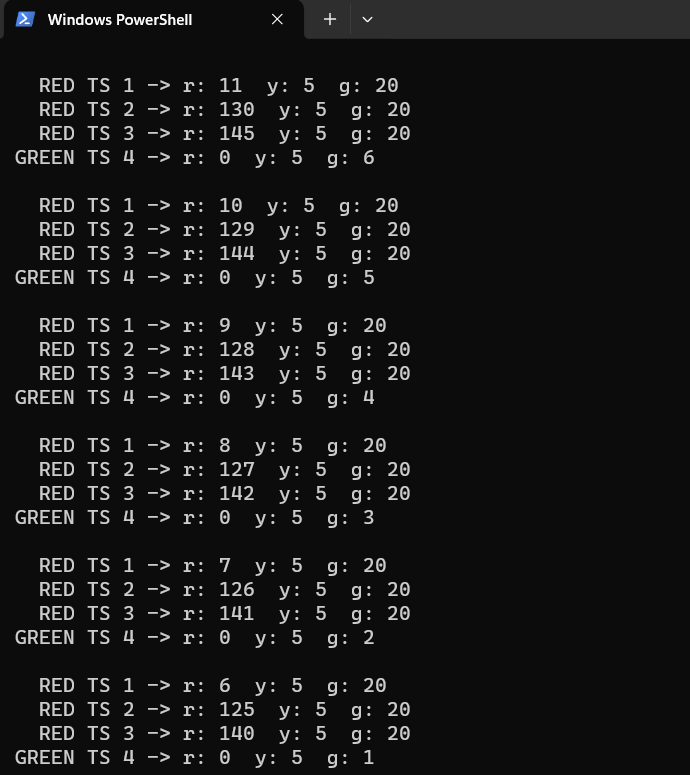
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test case #** | **Test case Name** | **Test case Description** | **Inputs** | **Expected Output** | **Actual Output** | **Status** |
| **1** | Normal traffic flow | To check whether the normal traffic cycle works or not. | Sending equal number of vehicles at each lane. | The duration of the green signal at each lane is same. | The duration of the green signal at each lane is same. | Pass |
| **2** | Camera connectivity | The connectivity of camera is tested to capture images or videos. | Inputting the program to capture the real time image or video. | Capture image or video. | Error. | Fail. |
| **3** | Executing the algorithm | Executing algorithm to detect the vehicles and classify them. | Image captured by camera. | Number of vehicles and value of delay of green signal. | Number of vehicles and value of delay of green signal. | Pass |
| **4** | Congestion control | Capturing the image using camera and sending to algorithm to detect vehicles. | Connect camera and send image to system. | Value of delay of green signal. | Value of delay of green signal. | Pass |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test case #** | **Test case Name** | **Test case Description** | **Inputs** | **Expected Output** | **Actual Output** |  |
| **5** | Emergency Vehicle detection | RFID tag is tested to detect the emergency vehicle. | Taping the RFID key on the board. | Turn on green signal for emergency vehicle immediately. | Turn on green signal for emergency  vehicle immediately. | Pass |
| **6** | Camera connectivity | The connectivity of camera is tested to capture images or videos. | Inputting the program to capture the real time image or video. | Capture image or video. | Error. | Fail. |
| **8** | SmartFlow testing | Congestion control and emergency vehicle detection is tested. | More number of vehicles at one lane and emergency vehicle on other lane. | Turn on green signal for emergency vehicle immediately. | Turn on green signal for emergency vehicle immediately. | Pass |

# Results and Discussion

**Congestion control**

****

****

# 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 and adjust green delay according to the number of vehicles and the emergency vehicle is detected using RFID tag so that it does not experience any traffic or delay in reaching hospital.

This project 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 smart traffic management system 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.

# Scope for Further Enhancement

For the proposed system controls the traffic congestion using camera and detect the emergency. For this system we can implement the accident detection and inform the nearby ambulance, also we can implement the Li-fi navigation so that tit will be easy for the ambulance driver to take the nearest rout to hospital with low traffic experience. Following the traffic rules id also important. We can implement the traffic rule violation detecting. The cameras will take the images of vehicles who violate the rules and by recognizing the plate number we can send message to the driver.