TRAFFIC MANAGEMENT WITH IOT

A

Mini Project Report

Submitted in partial fulfilment of the

Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

<K. ASISH><1602-20-737-126>

<R.MANJULA><1602-20-737-144>

<SRINIVAS><1602-20-737-315>

Department of Information Technology

Vasavi College of Engineering (Autonomous)

VASAVI COLLEGE OF ENGINEERING (AUTONOMOUS)

(AFFILIATED TO OSMANIA UNIVERSITY)

HYDERABAD - 500 030

Department of Information Technology

DECLARATION BY CANDIDATE

We, **<K.ASISH>, <R.MANJULA>, <SRINIVAS>,** bearing hall ticket number, **<1602-20-737-126>,<1602-20-737-144>,<1602-20-737-315>**
hereby declare that the project report entitled **<"TRAFFIC MANAGEMENT WITH IOT">** Department of Information Technology, Vasavi College of Engineering, Hyderabad, is submitted in partial fulfillment of the requirement for the award of the degree of **Bachelor of Engineering** in **Information Technology**

This is a record of bonafide work carried out by me and the results embodied in this project report has not been submitted to any other university or institute for the award of any other degree or diploma.

<K. ASISH><1602-20-737-1126>

<R.MANJULA><1602-20-737-144>

<SRINIVAS><1602-20-737-315>

VASAVI COLLEGE OF ENGINEERING (AUTONOMOUS)

(AFFILIATED TO OSMANIA UNIVERSITY)

HYDERABAD - 500 030

Department of Information Technology

BONAFIDE CERTIFICATE

This is to certify that the project entitled “**TRAFFIC MANAGEMENT WITH IOT**” being submitted by **ASISH, MANJULA, SRINIVAS**bearing **1602-20-737-126, 1602-20-737-144, 1602-20-737-315**, in partial fulfillment of the requirements for the completion of MINI PROJECT of Bachelor of Engineering in Information Technology is a record of bonafide work carried out by them under my guidance.

Internal Guide	External Examiner	Dr.K Ram Moahn Rao
Mrs.B Leelavathy		HOD, IT

ACKNOWLEDGEMENT

We thank the department of INFORMATION TECHNOLOGY, for introducing the subject “Mini Project-2” in BE fifth semester.

We would also like to show our appreciation to our Honorable principal, Dr S V Ramana sir, our HOD K. Ram Mohan Rao for supporting us and our mini project lecturer, Mrs Leelavathy mam, for letting us properly understand the process of doing a project and for providing valuable insight and expertise that has greatly assisted us in the making of the project.

TABLE OF CONTENTS:

ABSTRACT

- Traffic is the serious issue which each nation faces due to the expansion in number of vehicles.
- One of the strategies to beat the traffic issue is to build up a smart traffic control framework which depends on computing the traffic density and about utilizing constant video and picture preparing procedures.
- The topic is to control the traffic by deciding the traffic density on each roadside and control the traffic signal smartly by utilizing the density data.
- In this paper, an automated system based on processing of real time videos is proposed for detection of vehicles and recording count of them.
- The System will consist of various stages which includes Object Car Detection and Signal variation based on density.
- Captured video will be converted into frames and which will be pre-processed for object detection using Haar-Cascade than detected object count will be used to obtain the density and manipulate the signal accordingly.
- The density count algorithm works by contrasting the ongoing edge of live video by the reference picture and via looking through vehicles just in the district of intrigue (for example street region).
- The figured vehicle thickness can be contrasted and other course of the traffic so as to perform control of the traffic flags in more smart and proficient manner.

CHAPTER 1

INTRODUCTION

Traffic clogging has become a difficult issue in urban areas. The fundamental explanation is the increase in population in the urban area that along these lines there is high vehicular travel, which births clog issue. Because of traffic blockages there is high cost of transportation as a result of time wastage and additional fuel use.

Why traffic management?

- For example, in the event that there is a crisis vehicle with the basic patient ready. In that circumstance if an emergency vehicle stalls out in an overwhelming congested road, at that point there are high possibilities that the patient can't arrive at the clinic on schedule. So, it is critical to structure a keen traffic framework which controls traffic brilliantly to maintain a distance from accidents, crashes and roads turned parking lots
- The most well-known explanation of traffic blockage in underdeveloped nations is a traffic signal con- trolling which influences the traffic stream. For example, if one path has less traffic and the other path has more traffic yet the green light is same then it creates problem.
- By considering the above model if the path with higher traffic thickness should turn on the green signal light for a longer period than the lane with lesser density. It will solve the problem.

1.1 PURPOSE

1.To control traffic

2.To control waiting time of vehicles at signals

1.2 INTENDED AUDIENCE

The intended audience for this project is everyone who wants to know about Traffic Management Techniques.

Everyone with vehicles can use the technique to manage traffic clogging in their day today life.

1.Common people with vehicles.
2.VIPs.
3.Patient in ambulance.

1.3 PRODUCT SCOPE

This work reduces the waiting time for green light at traffic signal and counts the density of vehicles in lane.

1.4 PROBLEM DEFINATION

Signals are allocated a fixed time, the problem here is even if there are no vehicles in that lane the signal will turn green. we came up with a solution where signal lights will be manipulated according to the density of the vehicles in that particular lane. The lane with more vehicles will be given a preference. Previously implemented using sensors. We are trying with video processing. Results in low costs, no need of sensors.

CHAPTER 2

RELATED WORK –

we propose a density-based counting of vehicles which gives us exact information for signal decision making.

.

CHAPTER 3

PROPOSED WORK –

3.1 Steps--

- Image processing
- Image cropping
- RGB to grayscale transformation
- Threshold
- Contour
- Calculate traffic density
- Conventional traffic control system manual controlling
- Automatic controlling
- Hardware software requirements

3.2 UI prototypes or screenshots

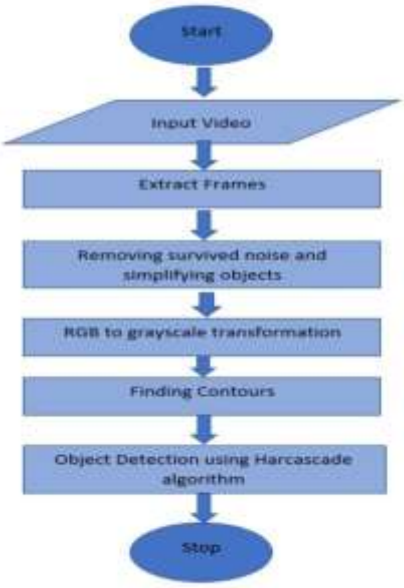

Start
Input Video
Extract Frames
Removing survived noise and simplifying objects
RGB to grayscale transformation
Finding Contours
Object Detection using Harcascade algorithm
Stop

Figure 1. Video processing

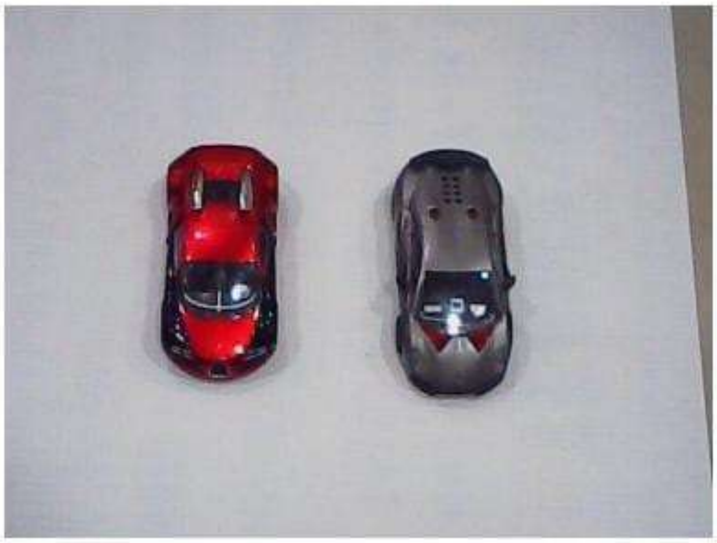


Figure 4. Image Capturing

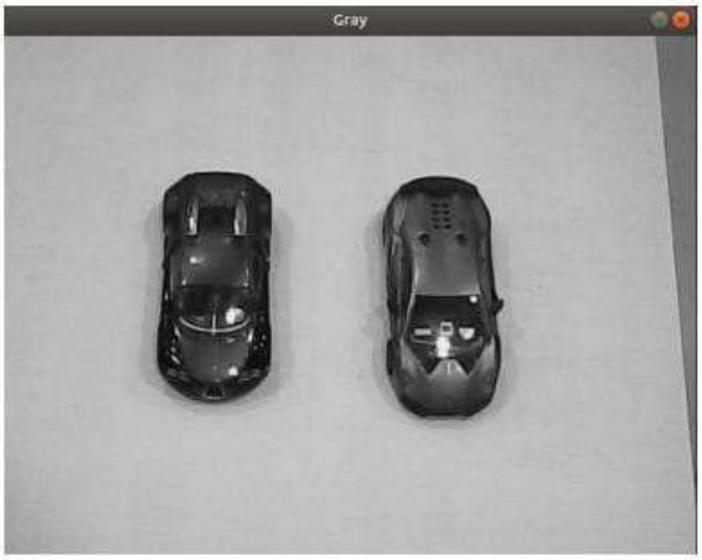
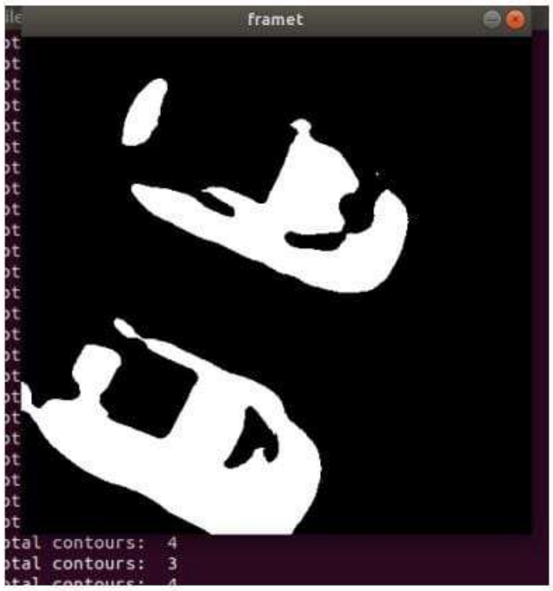
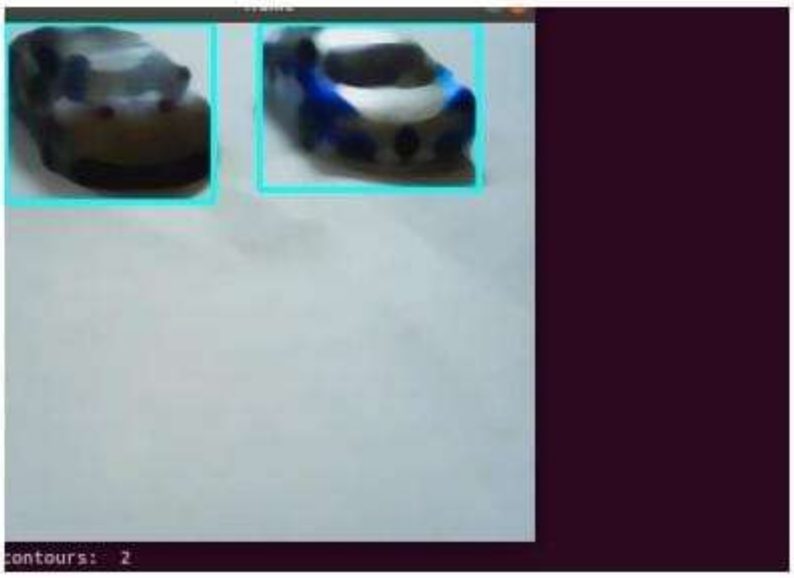


Figure 5. RGB to gray scale transformation



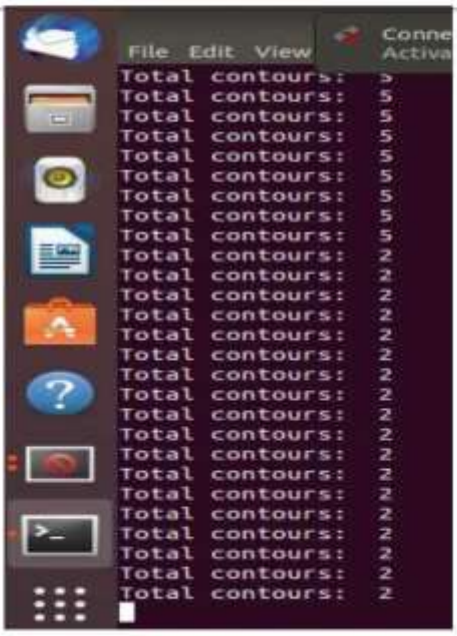
```
otal contours:  4
otal contours:  3
```

Figure 6. Threshold Image



```
contours:  2
```

Figure 7. Car Detection by Contours



```
Total contours:  5
Total contours:  5
Total contours:  5
Total contours:  5
Total contours:  5
Total contours:  5
Total contours:  5
Total contours:  5
Total contours:  5
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
```

Figure 8. Density Count

3.3 Architecture and Technology used –

Technology used –

- Arduino
- Python
- Open CV
- Pyfirmata

Architecture –

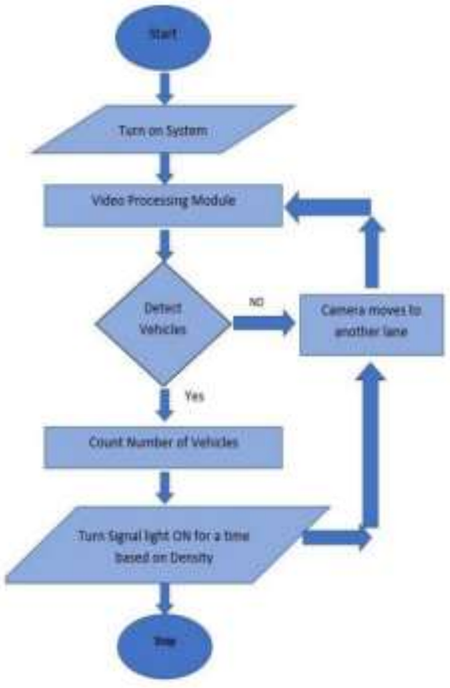


Figure 2. System Architecture

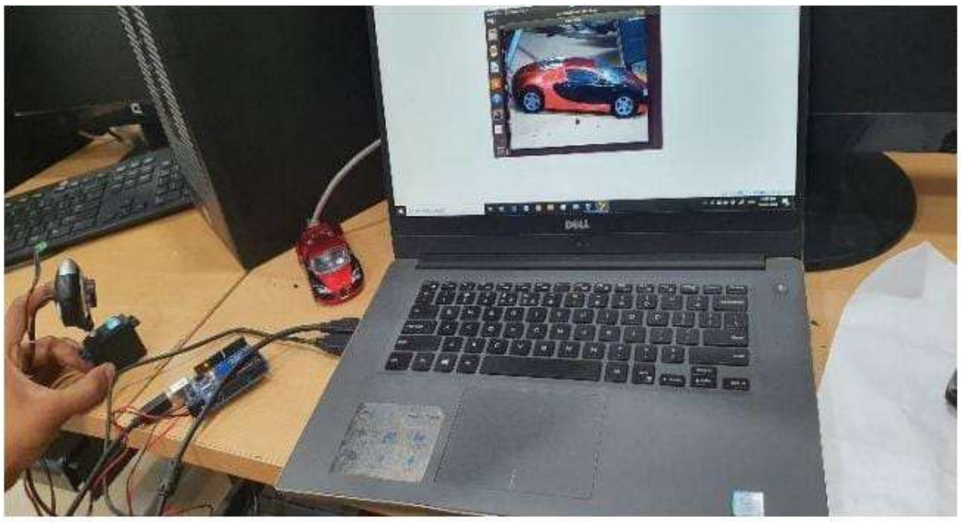
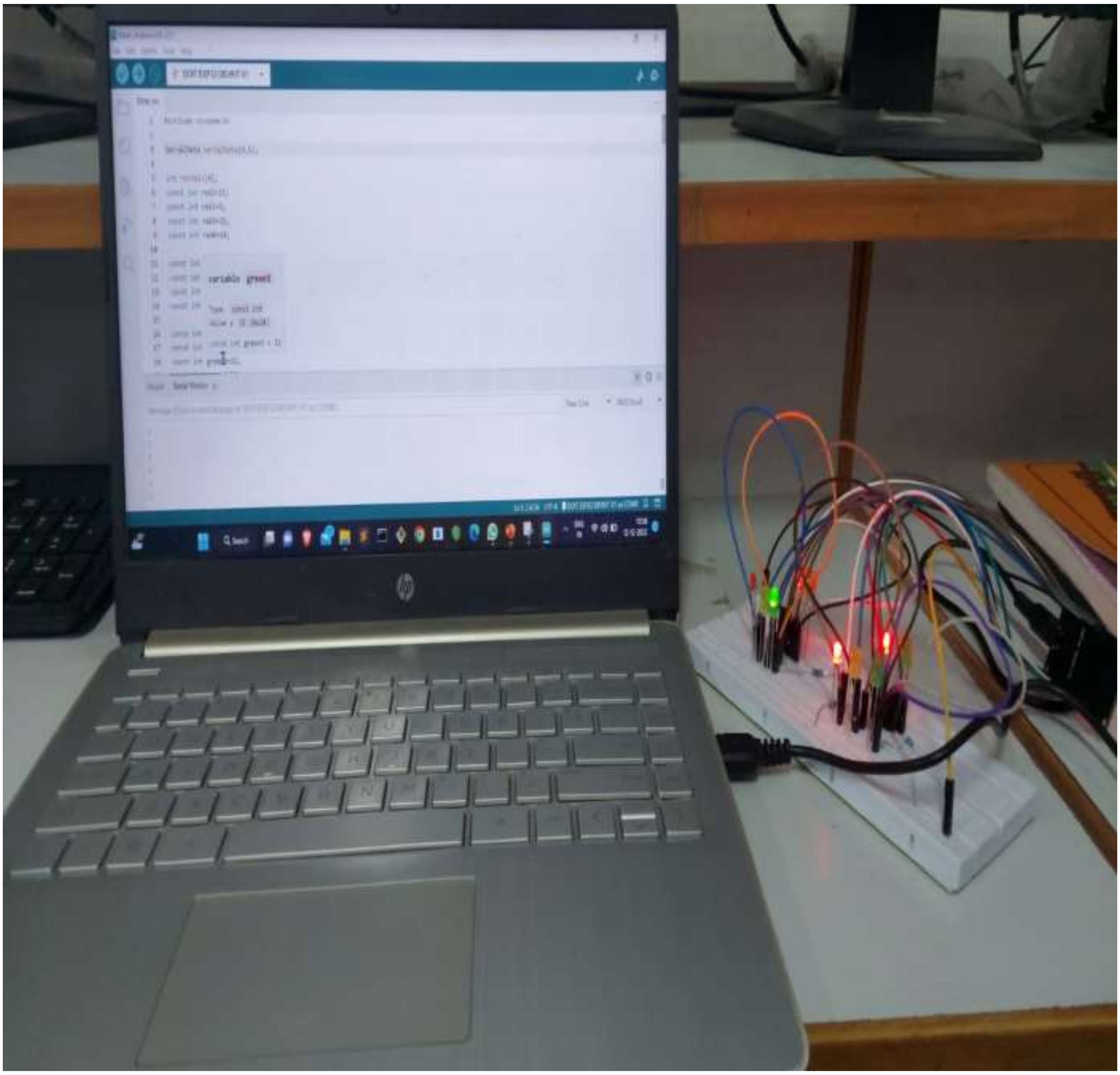


Figure 3. System Setup

3.4 Design –

This is the system setup and design used for our project



3.5 Implementation –

3.5.1 – Modules :

- OpenCV v3.4.9
- Image processing principle
- Python
- Arduino uno micro controller
- Techniques: blob detection and thresholding.
- Haar –Cascade

3.5.2 – Algorithm used:

1. Initially, we collected required data for video processing module in which the methods and algorithm is present to detect the vehicles.
2. This step applies a condition in which if it detects the vehicles has a yes or no answer with the preceding step to be done.
3. If the system does detect the vehicle then it will give the total number of count of vehicles in that lane using the video processing module
4. In this step once the count is given the signal turns green for the time specified according to the algorithm based on the density or count of the vehicles in that particular lane.
5. If the camera detects no vehicle on the particular lane that he is facing then the camera will move and change its direction facing to the other lane to count the vehicle density in that particular lane and repeat from step3.

Algorithm used are Open CV and video, image processing.

3.5.3 CODES--

```
#signals.py
import pyfirmata
import time
comp='COM11'
board = pyfirmata.Arduino(comp)
led_1=board.get_pin('d:2:o')
led_2=board.get_pin('d:3:o')
led_3=board.get_pin('d:4:o')
led_4=board.get_pin('d:5:o')
led_5=board.get_pin('d:6:o')
led_6=board.get_pin('d:7:o')
l1=[0,0]
def normaltime():
    return 2
def maxtime():
    return 5
def val1():
    return l1[0]
def val2():
    return l1[1]
def signal1():
    l1[0]=1
    l1[1]=0
    led_2.write(0)
    led_5.write(0)
    led_3.write(1)
    led_4.write(1)
def wait():
    l1[0]=0
    l1[1]=0
    led_2.write(1)
    led_3.write(0)
    led_4.write(0)
    led_5.write(1)
    led_1.write(0)
    led_6.write(0)
```

```
def signal2():
    l1[0]=0
    l1[1]=1
    led_2.write(0)
    led_6.write(1)
    led_1.write(1)
    led_5.write(0)
def led_norm():
    l1[0]=1
    l1[1]=0
    led_2.write(0)
    led_5.write(0)
    led_3.write(1)
    led_4.write(1)
    time.sleep(5)
    led_2.write(1)
    led_3.write(0)
    led_4.write(0)
    led_5.write(1)
    time.sleep(1)
    l1[0]=1
    l1[1]=0
    led_2.write(0)
    led_6.write(1)
    led_1.write(1)
    led_5.write(0)
    time.sleep(5)
    led_2.write(1)
    led_1.write(0)
    led_6.write(0)
    led_5.write(1)
    time.sleep(1)
```

```
def signalmax_1():
    l1[0]=1
    l1[1]=0
    led_2.write(0)
    led_5.write(0)
    led_3.write(1)
    led_4.write(1)
    time.sleep(15)
    led_2.write(1)
    led_3.write(0)
    led_4.write(0)
    led_5.write(1)
    time.sleep(1)
    l1[0]=1
    l1[1]=0
    led_2.write(0)
    led_6.write(1)
    led_1.write(1)
    led_5.write(0)
    time.sleep(15)
    led_2.write(1)
    led_1.write(0)
    led_6.write(0)
    led_5.write(1)
    time.sleep(1)
def signalmax_2():
    l1[0]=1
    l1[1]=0
    led_2.write(0)
    led_5.write(0)
    led_3.write(1)
    led_4.write(1)
    time.sleep(15)
    led_2.write(1)
    led_3.write(0)
    led_4.write(0)
    led_5.write(1)
    time.sleep(1)
    l1[0]=1
    l1[1]=0
    led_2.write(0)
    led_6.write(1)
    led_1.write(1)
    led_5.write(0)
    time.sleep(15)
    led_2.write(1)
    led_1.write(0)
    led_6.write(0)
    led_5.write(1)
    time.sleep(1)
```

```
#v_d.py
import cv2
from signals import *
import time
cascade_src = 'cars.xml'
video_src = 'dataset/video.avi'
video_src1 = 'dataset/video1.avi'
l=[0,0]
cap = cv2.VideoCapture("dataset/video.avi")
cap1 = cv2.VideoCapture("dataset/video1.avi")
car_cascade = cv2.CascadeClassifier(cascade_src)
ret, img = cap.read()
ret1,img1 = cap1.read()
num=0
def returnval():
    return l[:]
def fun1(time1):
    now=time.time()
    timer = 0
    while timer<=time1:
        ret, img = cap.read()
        ret1,img1 = cap1.read()
        if (type(img) == type(None)):
            break
        if(type(img1)==type(None)):
            break
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        gray1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
        #cv2.imshow('video', gray)
        cars = car_cascade.detectMultiScale(gray, 1.1, 1)
        cars1 = car_cascade.detectMultiScale(gray1, 1.1, 1)
        for (x,y,w,h) in cars:
            cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)
        for (x,y,w,h) in cars1:
            cv2.rectangle(img1,(x,y),(x+w,y+h),(0,0,255),2)
        cv2.imshow('video', img)
        cv2.imshow('video1', img1)
        if(val1()==0):
            l[0]=len(cars)
        if(val2()==0):
            l[1]=len(cars1)
        if cv2.waitKey(33) == 27:
            break
        end = time.time()
        timer = round(end-now)
        print(timer)
    cv2.destroyAllWindows()
        #return num
```

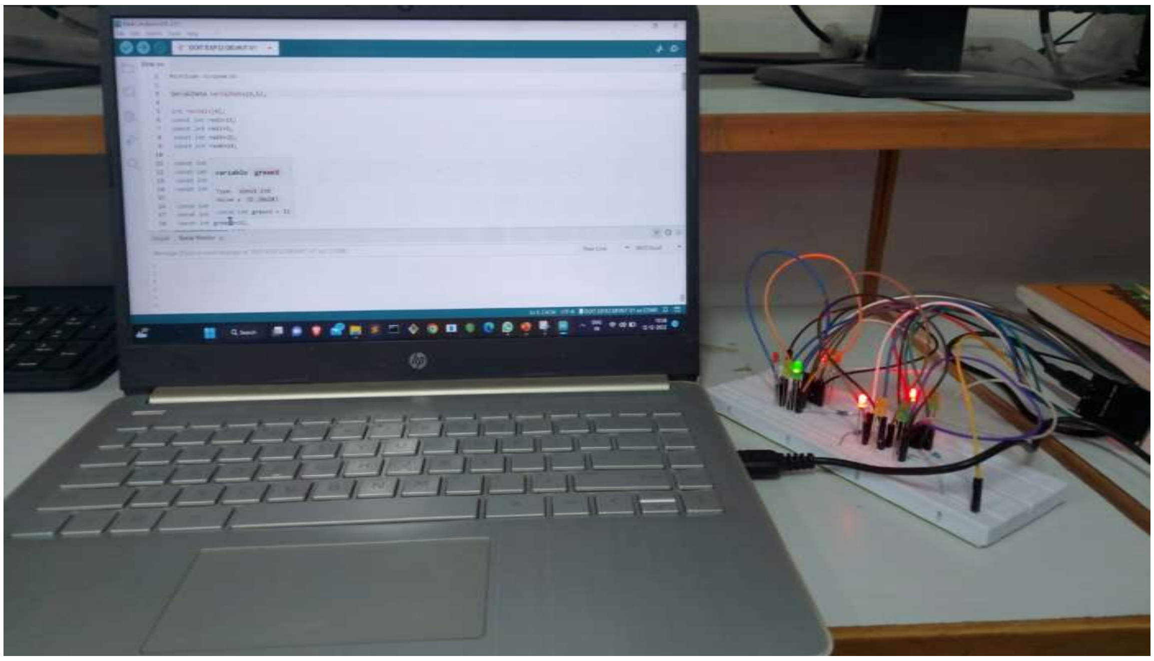
```
#main.py
from vehicle_detection import *
from signals import *
import time
count=1
preval=[]
while True:
    if(count<=2):
        signal1()
        fun1(normaltime())
        wait()
        time.sleep(1)
        signal2()
        fun1(normaltime())
        wait()
        time.sleep(1)
        temp=returnval()
        preval.append(temp)
        count+=1
        print(preval)
    else:
        dif=preval[0][0]-preval[1][0]
        if(dif>=3):
            signal1()
            fun1(maxtime())
        else:
            signal1()
            fun1(normaltime())
        wait()
        time.sleep(1)
        dif=preval[0][1]-preval[1][1]
        if(dif>=1):
            signal2()
            fun1(maxtime())
        else:
            signal2()
            fun1(normaltime())
        wait()
        time.sleep(1)
        preval.pop(0)
        preval.append(returnval())
        print(count)
        print(preval)
```

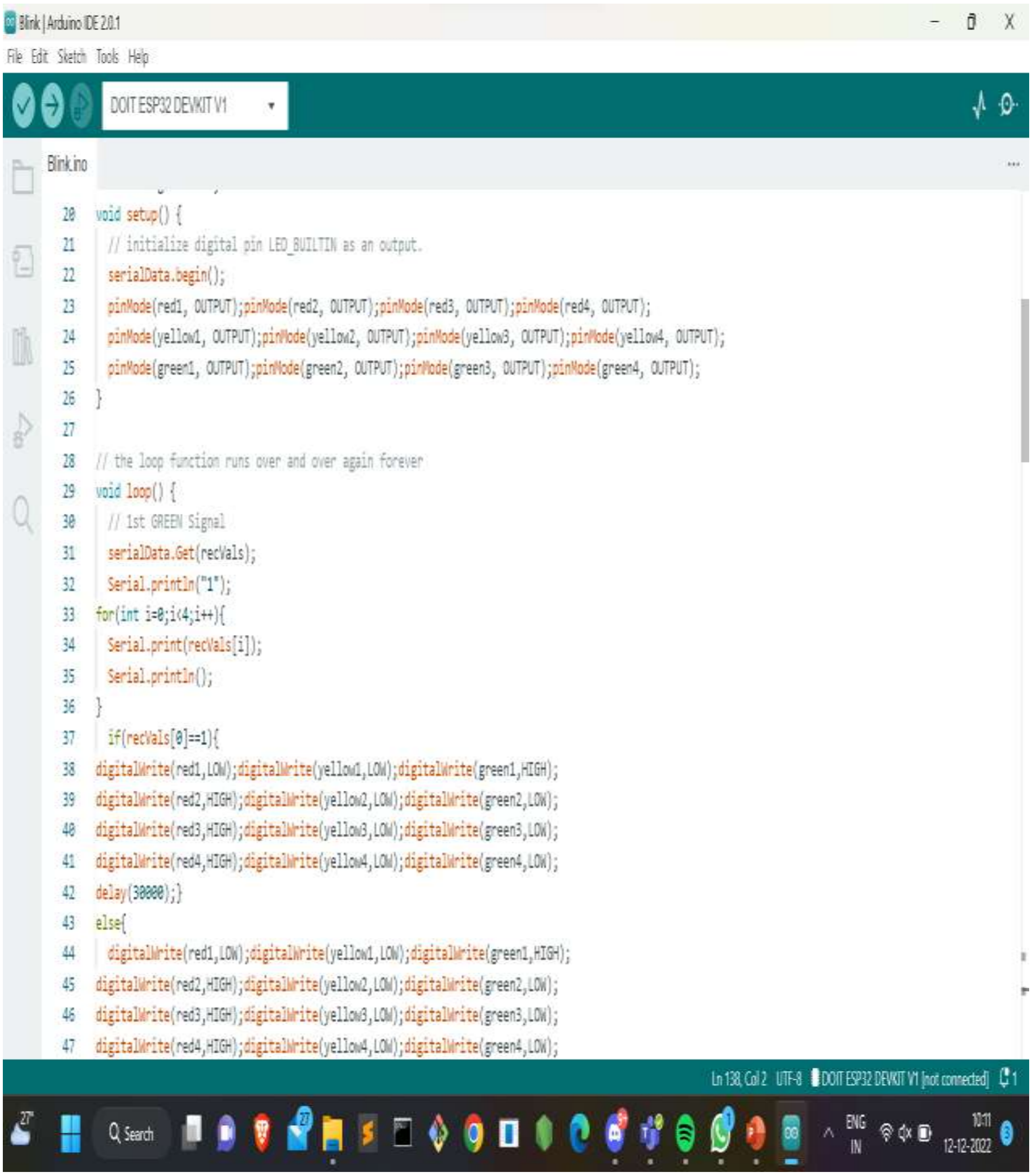
3.5.3 – GitHub Links –

https://github.com/manju-7/traffic-management-with-iot

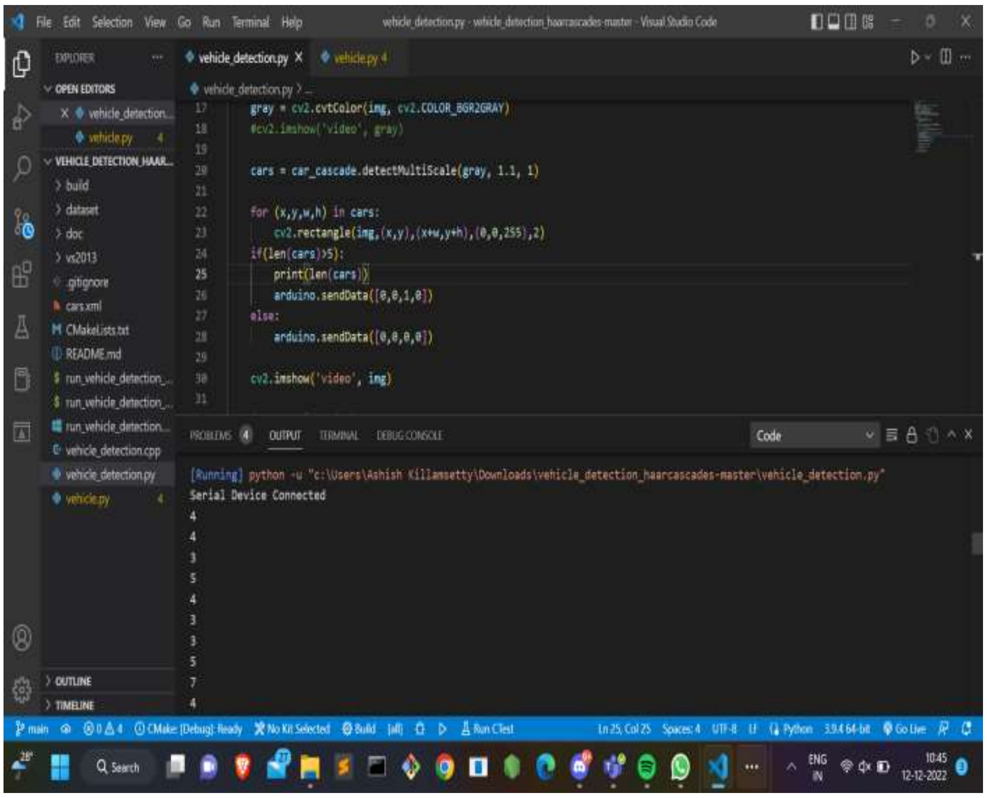
3.6 – Testing –

WE used Arduino and python for testing .





```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  serialData.begin();
  pinMode(red1, OUTPUT);pinMode(red2, OUTPUT);pinMode(red3, OUTPUT);pinMode(red4, OUTPUT);
  pinMode(yellow1, OUTPUT);pinMode(yellow2, OUTPUT);pinMode(yellow3, OUTPUT);pinMode(yellow4, OUTPUT);
  pinMode(green1, OUTPUT);pinMode(green2, OUTPUT);pinMode(green3, OUTPUT);pinMode(green4, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  // 1st GREEN Signal
  serialData.Get(recVals);
  Serial.println("1");
for(int i=0;i<4;i++){
  Serial.print(recVals[i]);
  Serial.println();
}
  if(recVals[0]==1){
digitalWrite(red1,LOW);digitalWrite(yellow1,LOW);digitalWrite(green1,HIGH);
digitalWrite(red2,HIGH);digitalWrite(yellow2,LOW);digitalWrite(green2,LOW);
digitalWrite(red3,HIGH);digitalWrite(yellow3,LOW);digitalWrite(green3,LOW);
digitalWrite(red4,HIGH);digitalWrite(yellow4,LOW);digitalWrite(green4,LOW);
delay(30000);}
else{
  digitalWrite(red1,LOW);digitalWrite(yellow1,LOW);digitalWrite(green1,HIGH);
digitalWrite(red2,HIGH);digitalWrite(yellow2,LOW);digitalWrite(green2,LOW);
digitalWrite(red3,HIGH);digitalWrite(yellow3,LOW);digitalWrite(green3,LOW);
digitalWrite(red4,HIGH);digitalWrite(yellow4,LOW);digitalWrite(green4,LOW);
```



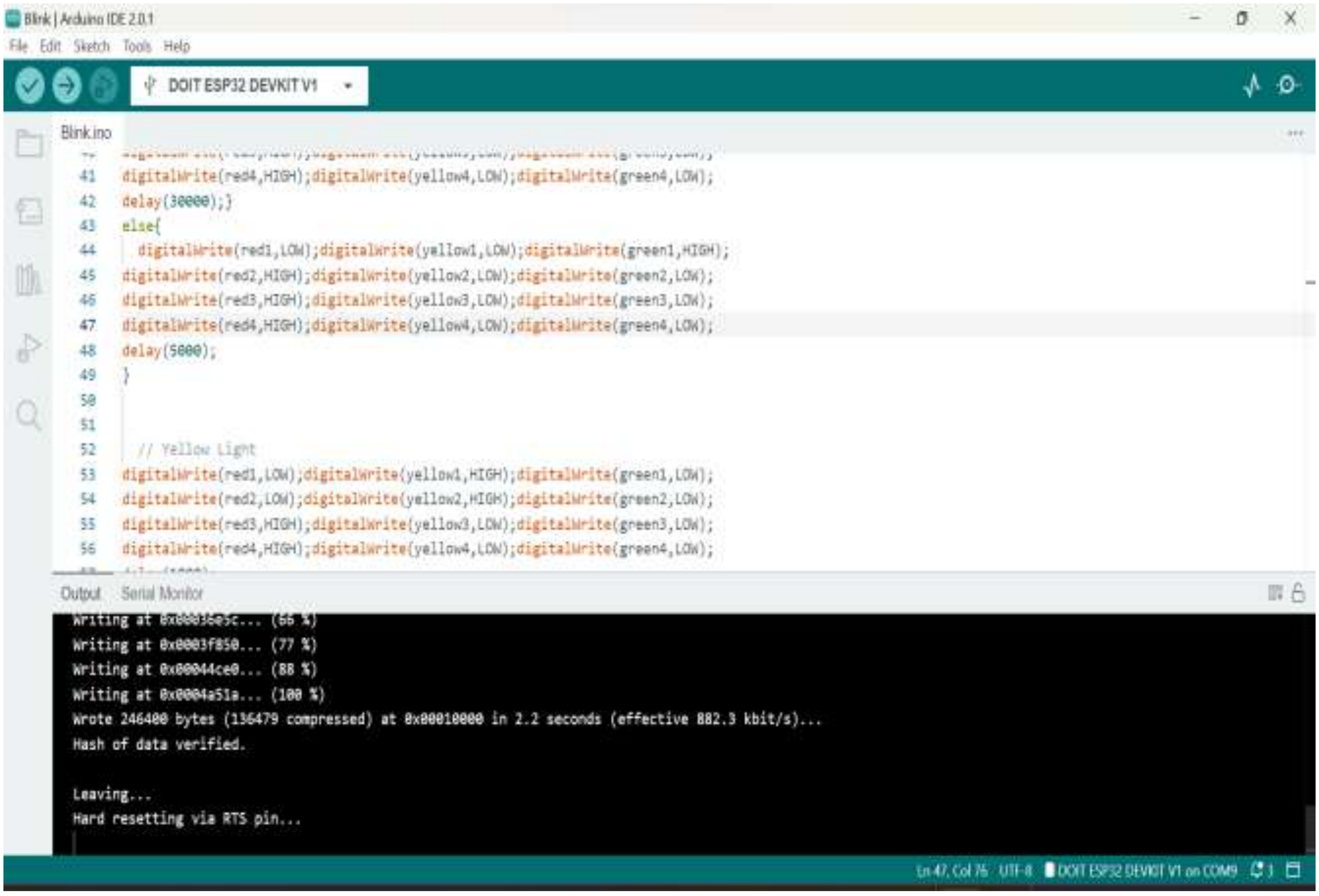
```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
#cv2.imshow('video', gray)

cars = car_cascade.detectMultiScale(gray, 1.1, 1)

for (x,y,w,h) in cars:
    cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)
if(len(cars)>5):
    print(len(cars))
    arduino.sendData([0,0,1,0])
else:
    arduino.sendData([0,0,0,0])

cv2.imshow('video', img)
```

```
[Running] python -u "c:\Users\Ashish Killamsetty\Downloads\vehicle_detection_haarcascades-master\vehicle_detection.py"
Serial Device Connected
4
4
3
5
4
3
3
5
7
4
```

CHAPTER 4 –

RESULTS

The following are the results obtained after implementation –


Blink | Arduino IDE 2.0.1

DOIT ESP32 DEVKIT V1

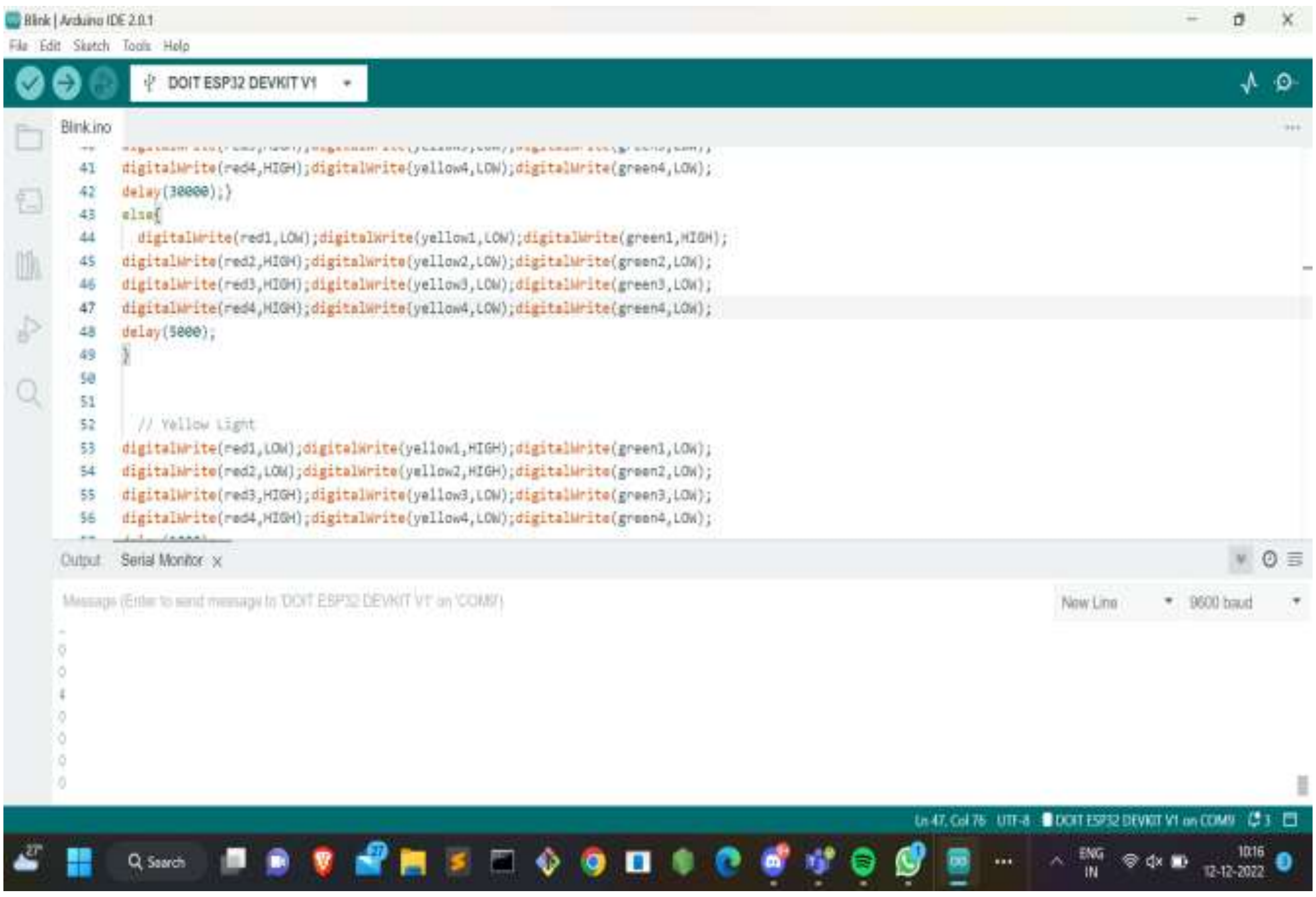
Blink.ino

```
digitalWrite(red4,HIGH);digitalWrite(yellow4,LOW);digitalWrite(green4,LOW);
delay(30000);}
else{
  digitalWrite(red1,LOW);digitalWrite(yellow1,LOW);digitalWrite(green1,HIGH);
digitalWrite(red2,HIGH);digitalWrite(yellow2,LOW);digitalWrite(green2,LOW);
digitalWrite(red3,HIGH);digitalWrite(yellow3,LOW);digitalWrite(green3,LOW);
digitalWrite(red4,HIGH);digitalWrite(yellow4,LOW);digitalWrite(green4,LOW);
delay(5000);
}


  // Yellow Light
digitalWrite(red1,LOW);digitalWrite(yellow1,HIGH);digitalWrite(green1,LOW);
digitalWrite(red2,LOW);digitalWrite(yellow2,HIGH);digitalWrite(green2,LOW);
digitalWrite(red3,HIGH);digitalWrite(yellow3,LOW);digitalWrite(green3,LOW);
digitalWrite(red4,HIGH);digitalWrite(yellow4,LOW);digitalWrite(green4,LOW);
```

Output   Serial Monitor

```
Writing at 0x00036e5c... (66 %)
Writing at 0x0003f850... (77 %)
Writing at 0x00044ce0... (88 %)
Writing at 0x0004a51a... (100 %)
Wrote 246400 bytes (136479 compressed) at 0x00010000 in 2.2 seconds (effective 882.3 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

Ln 47, Col 76   UTF-8   DOIT ESP32 DEVKIT V1 on COM9


Blink | Arduino IDE 2.0.1

DOIT ESP32 DEVKIT V1

Blink.ino

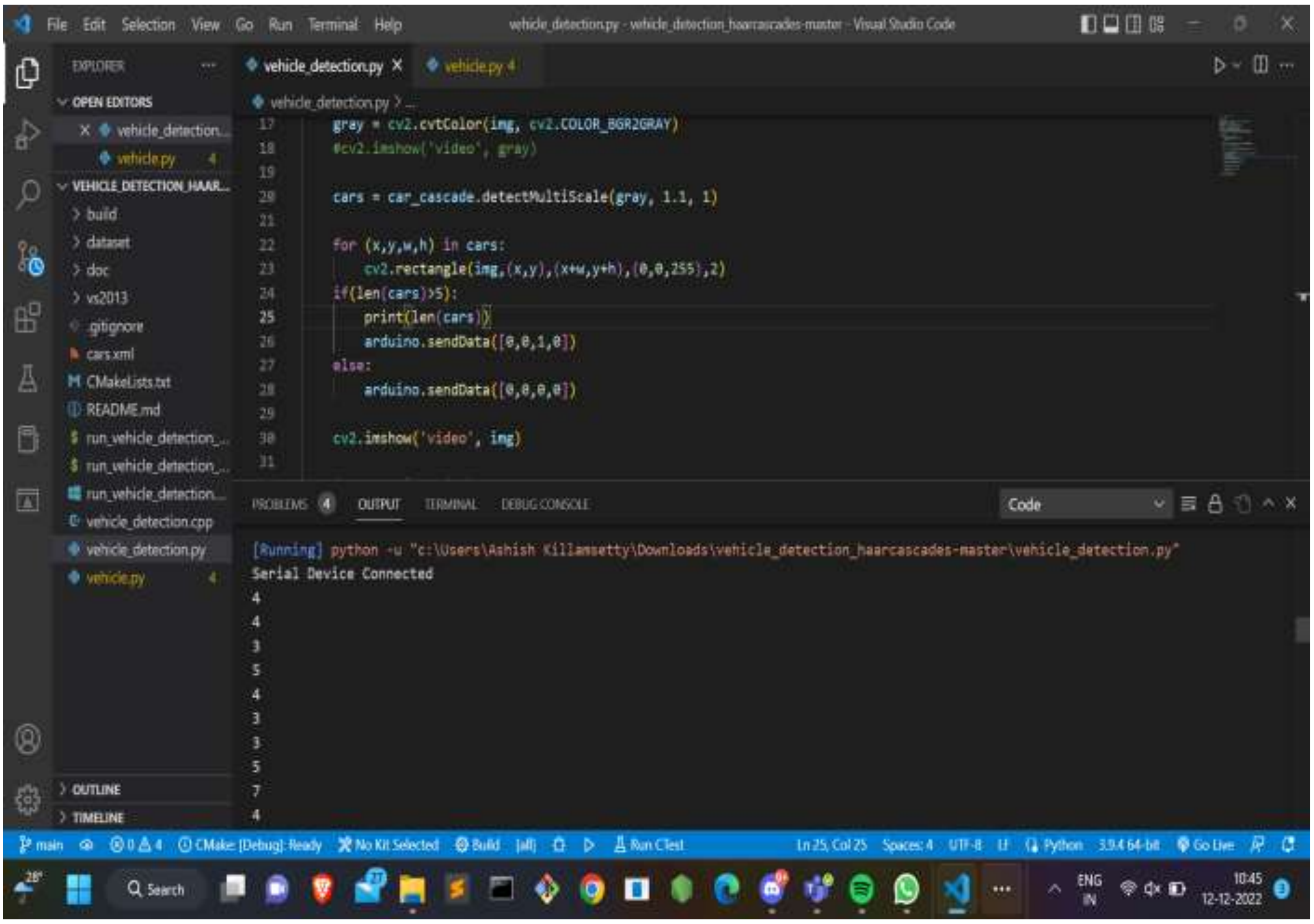
```
digitalWrite(red4,HIGH);digitalWrite(yellow4,LOW);digitalWrite(green4,LOW);
delay(30000);}
else{
  digitalWrite(red1,LOW);digitalWrite(yellow1,LOW);digitalWrite(green1,HIGH);
digitalWrite(red2,HIGH);digitalWrite(yellow2,LOW);digitalWrite(green2,LOW);
digitalWrite(red3,HIGH);digitalWrite(yellow3,LOW);digitalWrite(green3,LOW);
digitalWrite(red4,HIGH);digitalWrite(yellow4,LOW);digitalWrite(green4,LOW);
delay(5000);
}


  // Yellow Light
digitalWrite(red1,LOW);digitalWrite(yellow1,HIGH);digitalWrite(green1,LOW);
digitalWrite(red2,LOW);digitalWrite(yellow2,HIGH);digitalWrite(green2,LOW);
digitalWrite(red3,HIGH);digitalWrite(yellow3,LOW);digitalWrite(green3,LOW);
digitalWrite(red4,HIGH);digitalWrite(yellow4,LOW);digitalWrite(green4,LOW);
```

Output   Serial Monitor

Message (Enter to send message to 'DOIT ESP32 DEVKIT V1' on 'COM9')   New Line   9600 baud

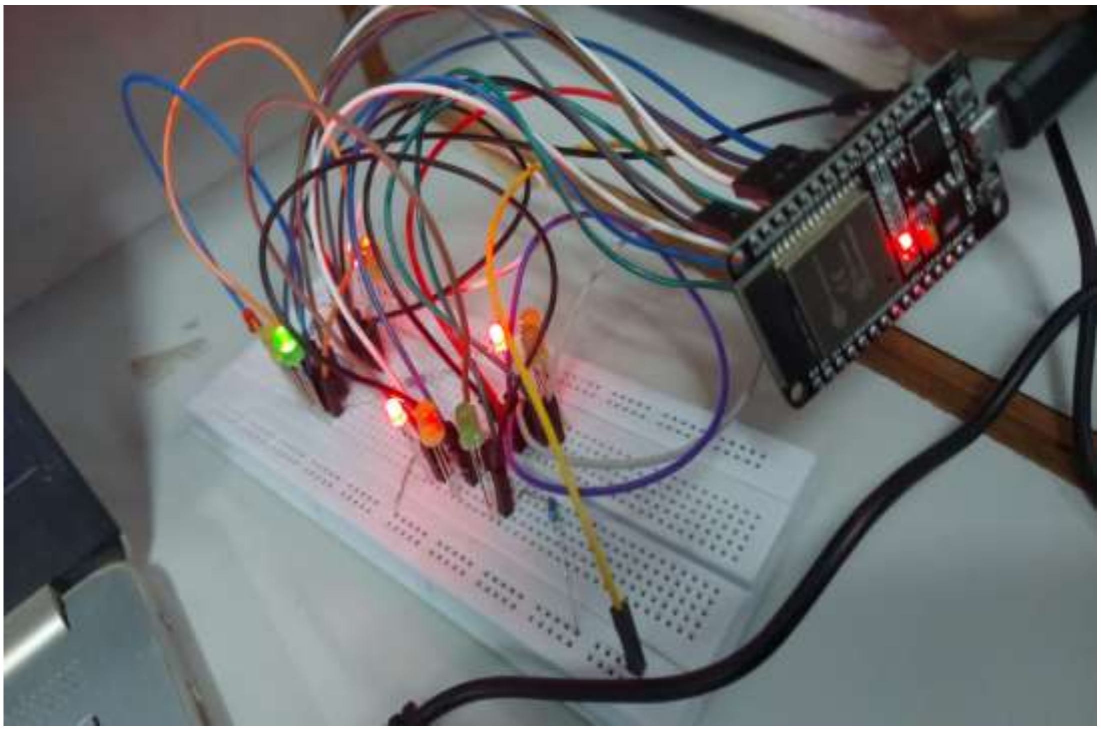
```
-
0
0
4
0
0
0
0
```

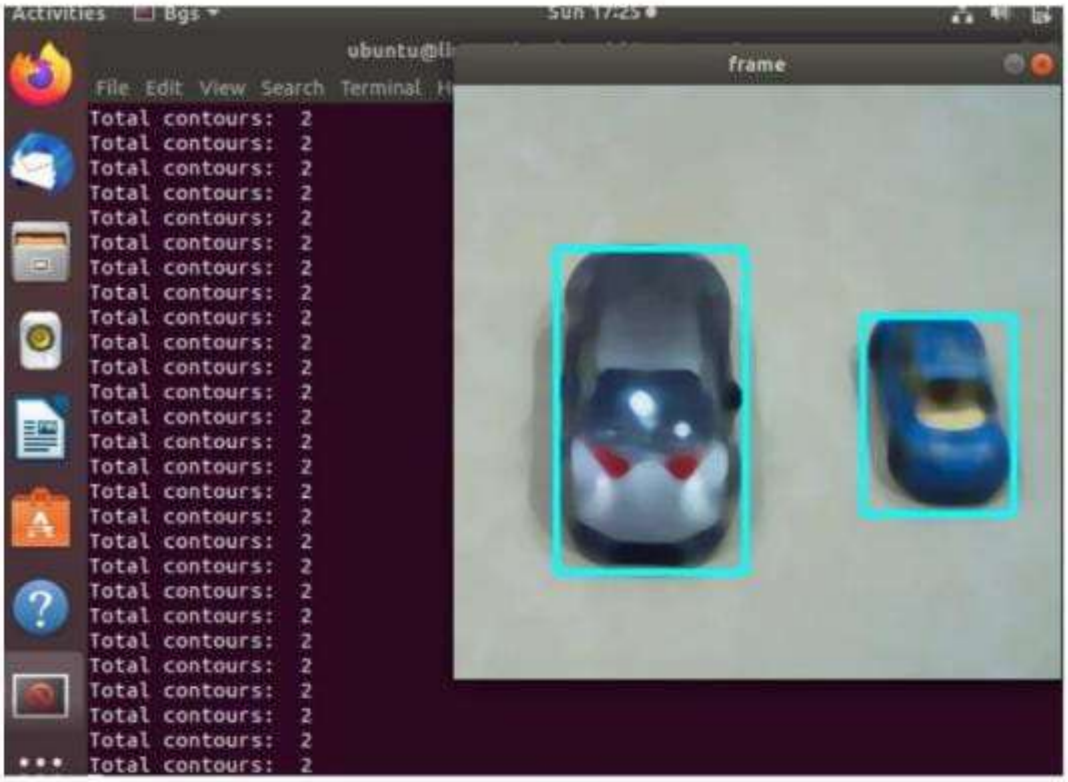
Ln 47, Col 76   UTF-8   DOIT ESP32 DEVKIT V1 on COM9



```
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    #cv2.imshow('video', gray)

    cars = car_cascade.detectMultiScale(gray, 1.1, 1)

    for (x,y,w,h) in cars:
        cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)
    if(len(cars)>5):
        print(len(cars))
        arduino.sendData([0,0,1,0])
    else:
        arduino.sendData([0,0,0,0])

    cv2.imshow('video', img)
```

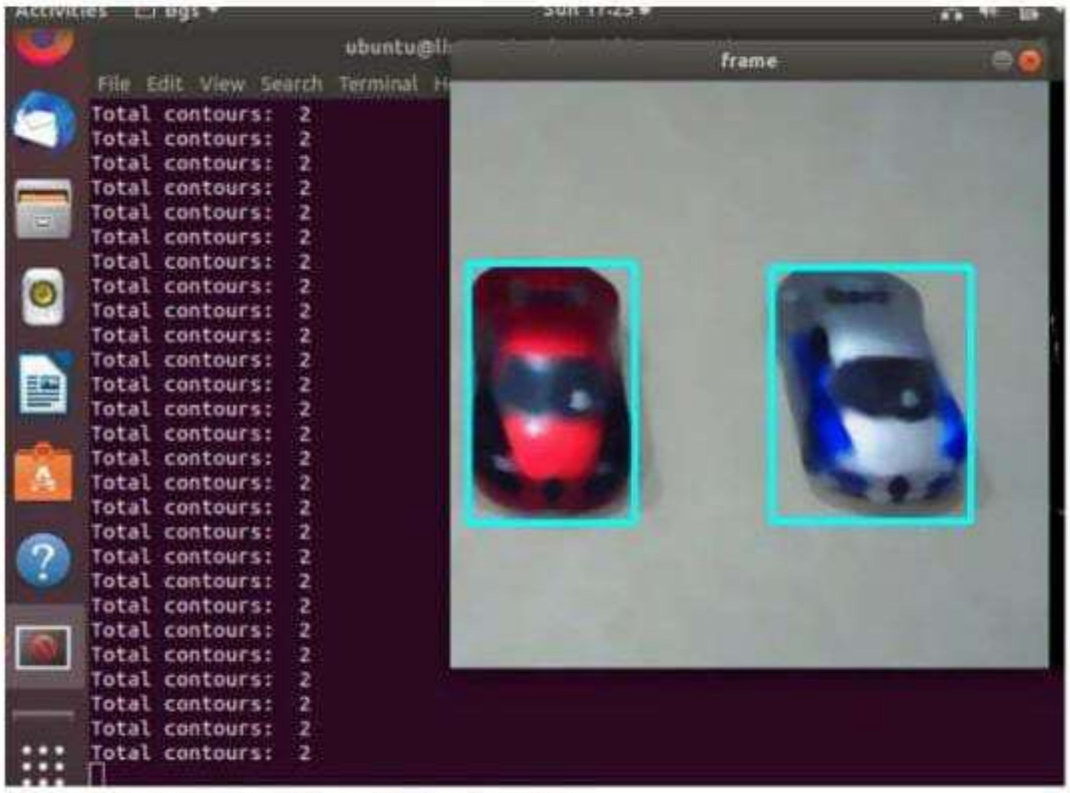
```
[Running] python -u "c:\Users\Ashish Killamsetty\Downloads\vehicle_detection_haarcascades-master\vehicle_detection.py"
Serial Device Connected
4
4
3
5
4
3
3
5
7
4
```





```
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
```

Figure 9a. Lane 1 vehicle density



```
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
Total contours:  2
```

Figure 9b. Lane 2 vehicle density

CHAPTER 5

DISCUSSION AND FUTURE WORK –

. This work can be upgraded further by proposing a framework for controlling the traffic density. That will decrease our serious issue of everyday life, traffic jam.

CHAPTER 6

CONCLUSION

Video processing is a good technique to control road congestion.
Consistent in detecting vehicle presence.
Using video processing and Object Detection mechanism, the proposed system achieves good accuracy in identifying the objects and estimating lane density.
Based on lane density the traffic issue can be resolve to greater extent

CHAPTER 7

REFERENCES –

https://www.sciencedirect.com/science/article/pii/S1110016815002033
https://www.researchgate.net/publication/334155445_Adaptive_Traffic_Lights_through_Traffic_Density_Calculation_on_Road_Pattern