SNIST
Sreenidhi Institute of Science and Technology
Because, Life is all about taking the right decisions
AUTONOMOUS

Real Time Traffic Density Monitoring Using Intelligent Traffic Signaling System

Aditya Raj Kothapally 17311A0529
M M Sai Prakash 17311A0438
M Vaishnavi 17311A0439
Vennam Roopshika 17311A1995
Under the Guidance of
Dr. Aruna Varanasi

CONTENTS

INTRODUCTION

- With the rapid development of road infrastructure, the volume of vehicles on the road network increases which leads to traffic Congestion.
- This is mainly caused due to the high up rise in the number of vehicles in a short span of time. To overcome such impact of traffic congestions, it is required to develop a Traffic control system depending on the density of vehicles.
- The proposed system would be based on the measurement of the actual traffic density on each of the 4 junctions at a traffic signal. For this implementation, we use Image Processing based techniques for calculating the density of vehicles on the lanes and set timer for the traffic lights accordingly using Rasberry Pi 3 or NvidiaJetson Nano an onboard powerful microprocessor.

- This system stands out from traditional traffic signaling system as it uses real time traffic data to calculate cars density and allocate respective timer to the signal.
- Apart from that, the timer allocated to the signal is dynamically updated after some pre-defined interval of time depending on the change in density of the cars at the signal.
- The system will be processed on an SOC and can have regular updates and bug fixes both remotely and hands on hence it is cognitive. Because of no use of any remote processing unit we do not have any latency due to communication lag and hence we need not rely on internet speed for better accuracy.
- The system is independent and it can also be contented to internet to store data remotely for survey or research purposes.

METHODOLOGY

- We can achieve the proposed system using a real time video and image processing techniques.
- In this system, the images of the four sides of the road are captured and the vehicle density is calculated using the harrcascade.
- The timer of signal will be increased dynamically as the traffic density increases.
- This density is used to sort the sides of the roads with the respect to the highest density among the 4 and evaluate its timer for the green signals which can last for maximum of 80 seconds using Rasberry Pi 3 or Nvidia Jetson Nano.
- The traffic on the sides of the roads are released in this sorted order of the vehicle density.
- Apart from that, the timer allocated to the signal is dynamically updated after some pre-defined interval of time depending on the change in density of the cars at the signal.
- After a round of all the roads receiving the green signal, the images of the 4 sides od the roads are again captured and it continues.

HARDWARE AND SOFTWARE REQUIREMENTS

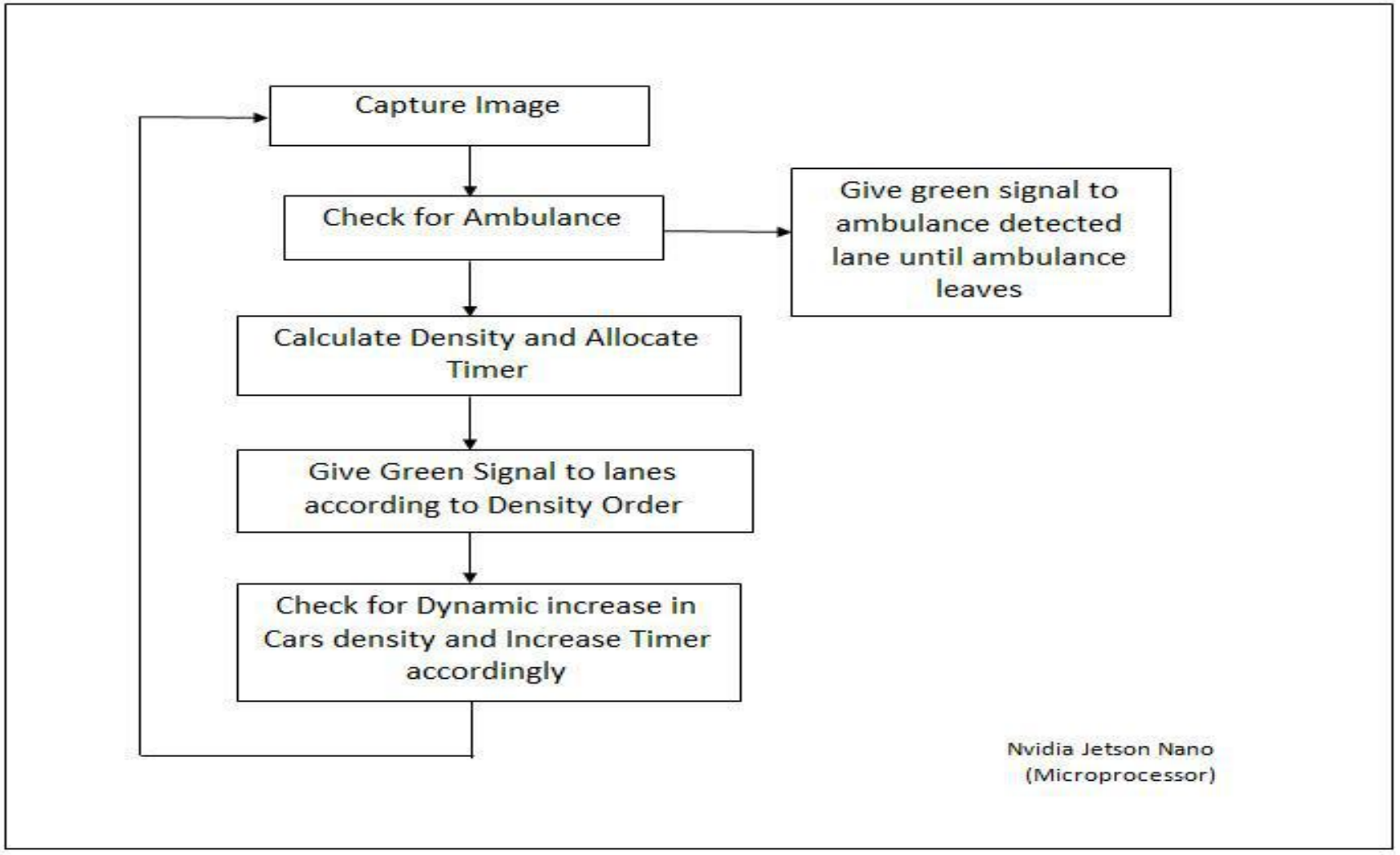
- **Hardware:**

  Rasberry Pi 3 or Nvidia Jetson Nano an onboard powerful microprocessor.

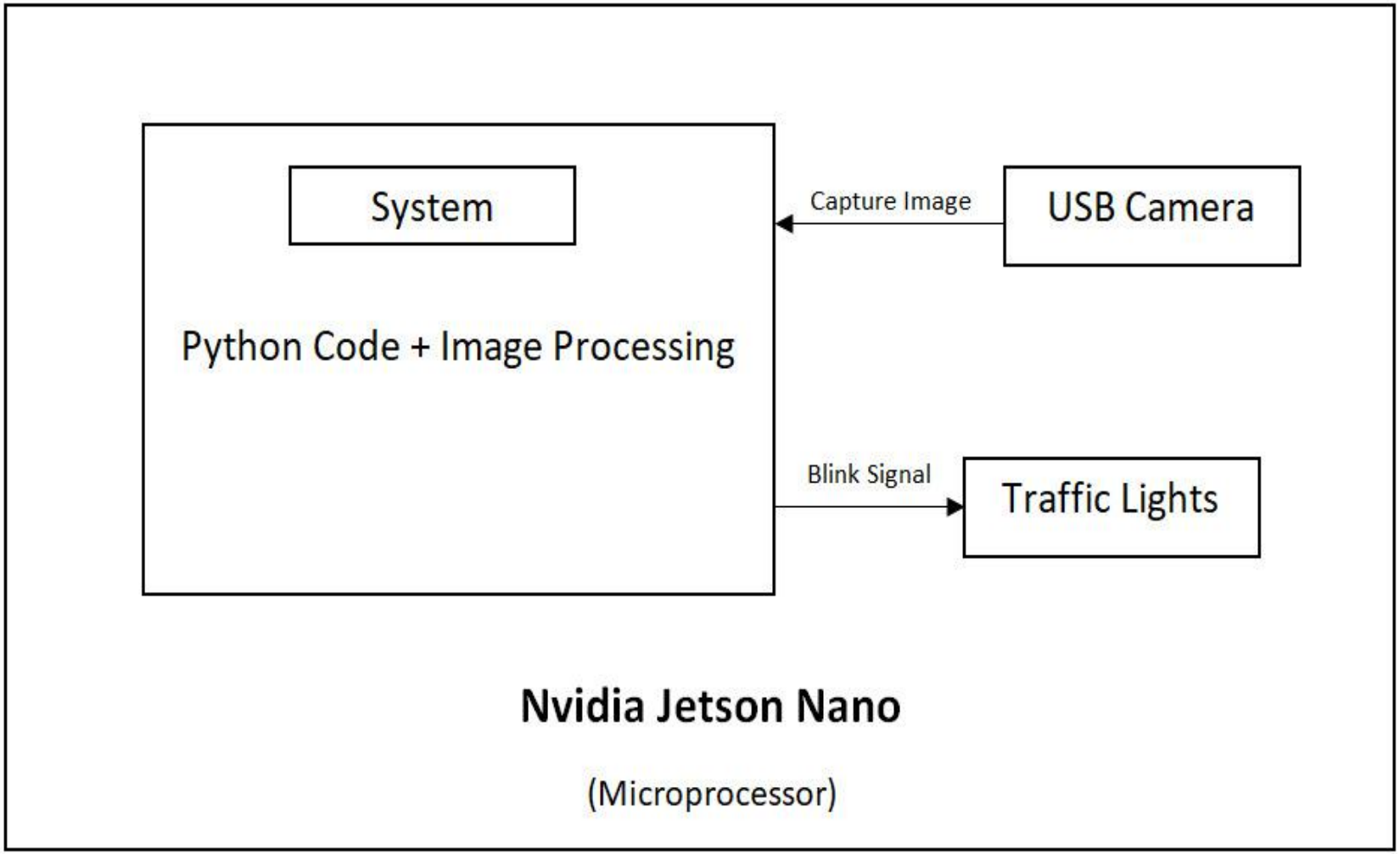
- **Software:**

  Python 2 and above.

Project flow



Architecture



Data Used

Python Code – Cars Detection

```
# Load Yolo
net = cv2.dnn.readNet("weights/yolov3.weights", "cfg/yolov3.cfg")
classes = []
with open("coco.names", "r") as f:
    classes = [line.strip() for line in f.readlines()]
layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]

font = cv2.FONT_HERSHEY_PLAIN

def detect(i):
    low_red = np.array([166, 155, 84])
    high_red = np.array([179,255,255])
    radius=0
    count=0
    img=cv2.imread(str(i)+".jpg")
    frame=cv2.resize(img,(960,540))
    height, width, channels = frame.shape

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    ret , thrash = cv2.threshold(gray, 240 , 255, cv2.CHAIN_APPROX_NONE)
    contours , hierarchy = cv2.findContours(thrash, cv2.RETR_TREE,
                                            cv2.CHAIN_APPROX_NONE)

    blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0),
                                 True, crop=False)
    net.setInput(blob)
    outs = net.forward(output_layers)

    # Showing informations on the screen
    class_ids = []
    confidences = []
    boxes = []
```

```
for out in outs:
    for detection in out:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > 0.1:
            # Object detected
            center_x = int(detection[0] * width)
            center_y = int(detection[1] * height)
            w = int(detection[2] * width)
            h = int(detection[3] * height)

            # Rectangle coordinates
            x = int(center_x - w / 2)
            y = int(center_y - h / 2)

            boxes.append([x, y, w, h])
            confidences.append(float(confidence))
            class_ids.append(class_id)

indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.8, 0.3)

for i in range(len(boxes)):
    if i in indexes:
        x, y, w, h = boxes[i]
        roi=frame[y:y+h, x:x+w]
        label = str(classes[class_ids[i]])
        confidence = confidences[i]
        color = (255,255,255)
        cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)
        cv2.putText(frame, label,
                    (x, y + 30), font, 1, color, 1)
        #print(label)
        if label=="car" or label=="truck" or label=="motorcycle":
            count=count+1
```

Python Code – Ambulance Detection

```
for contour in contours:
    approx = cv2.approxPolyDP(contour,
                              0.01* cv2.arcLength(contour, True), True)
    if len(approx) == 12:
        #cv2.drawContours(frame, [approx], 0, (0, 0, 0), 5)
        x = approx.ravel()[0]
        y = approx.ravel()[1] - 5
        col=frame[y:y+200,x:x+200]
        #print(col.size)
        if col.size!=0:
            #cv2.imwrite("dekh.jpg",col)
            hsv_frame = cv2.cvtColor(col, cv2.COLOR_BGR2HSV)
            blurred = cv2.GaussianBlur(col, (11, 11), 0)
            hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)
            mask = cv2.inRange(hsv_frame, low_red, high_red)
            kernel = np.ones((9,9),np.uint8)
            mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)
            mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)
            cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
                    cv2.CHAIN_APPROX_SIMPLE)[-2]
            if len(cnts)>0:
                c = max(cnts, key=cv2.contourArea)
                ((a,b), radius) = cv2.minEnclosingCircle(c)
        ##    red = cv2.bitwise_and(col, col, mask=mask)
        ##    cv2.imshow("Red",red)

        if radius>0.5:
            #imS = cv2.resize(frame, (960, 540))
            #cv2.putText(imS,"Ambulance", (x,y), font,0.5, (0,0,0))
            print("ambulance")
            #cv2.imwrite("dekh"+str(i)+".jpg",imS)
            #cv2.imshow('Emergency', imS)
            #cv2.waitKey(5000)
            #cv2.destroyAllWindows()
            return 108,100;
```

Python Code – Dynamic Signaling

```
for ii in range(4):
    #i=5
    if str(ii) in s:
        i=int(input("Enter Initial for already green lane no "+str(ii)+": "))
        a[ii][0],a[ii][1] = detect(i)
        while a[ii][0]==108:                #ambulance detection
            print("green for "+str(ii)+" because ambulance detected")
            time.sleep(1)
            i=int(input("Enter Ambulance lane dynamic: "))
            a[ii][0],a[ii][1] = detect(i)
            if a[ii][0]!=108:
                a[ii][0]=-1
                a[ii][1]=0
                print("green blinked at lane "+str(ii)+" for ambulance clearance")
        a[ii][0]=0
        a[ii][1]=0
    else:
        i=int(input("Enter Initial for lane no "+str(ii)+": "))
        a[ii][0],a[ii][1] = detect(i)
        while a[ii][0]==108:                #ambulance detection
            print("green for "+str(ii)+" because ambulance detected")
            time.sleep(1)
            i=int(input("Enter Ambulance lane dynamic: "))
            a[ii][0],a[ii][1] = detect(i)
            if a[ii][0]!=108:
                a[ii][0]=-1
                a[ii][1]=0
                s +=str(ii)
                print("green blinked at lane "+str(ii)+" for ambulance clearance")
                print("Lanes given Green -> "+s+" <-")            #appending to stirng because green given to this lane for ambulance clearance
                j -=1                   #decrementing j because green given to this lane for ambulance clearance
        print("time = "+str(a[ii][1]))
        print("count = "+str(a[ii][0]))
if j<0: break                           #checking beacuse irregularity may occur because of previous ambulance detection
#print(a)
```

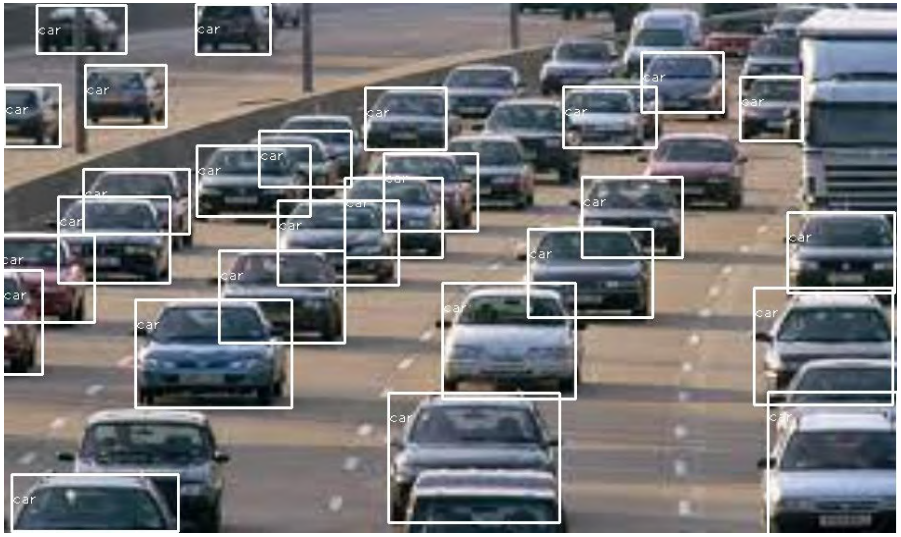
```
r=a[aa[3]][1]
s +=str(aa[3])
print("Lanes given Green -> "+s+" <-")
tim=r
tl=r
while tim<8:
    time.sleep(1)
    r=r-1
    ii=int(input("Enter dynamic:"))
    #ii=7
    c, t = detect(ii)
    while c==108:               #ambulance detection
        print("green for "+str(aa[3])+" because ambulance detected")
        time.sleep(1)
        i=int(input("Enter Ambulance lane dynamic: "))
        c, t = detect(i)
        if c!=108:
            c=-1
            t=0
            r=0
    if c>=10:
        tim=tim+t
        if tim>8:
            tim=8
        r=r+(tim-tl)
        tl=tim
        print("r="+str(r))
        print("time = "+str(tim))
    else:
        break
time.sleep(r)
if c!=-1: print("green blinked at lane "+str(aa[3])+" for "+str(tim)+" seconds")
else: print("green blinked at lane "+str(aa[3])+" for ambulance clearance")
print("red for "+str(aa[3]))
j-=1
#print("Lanes given Green -> "+s+" <-")
```

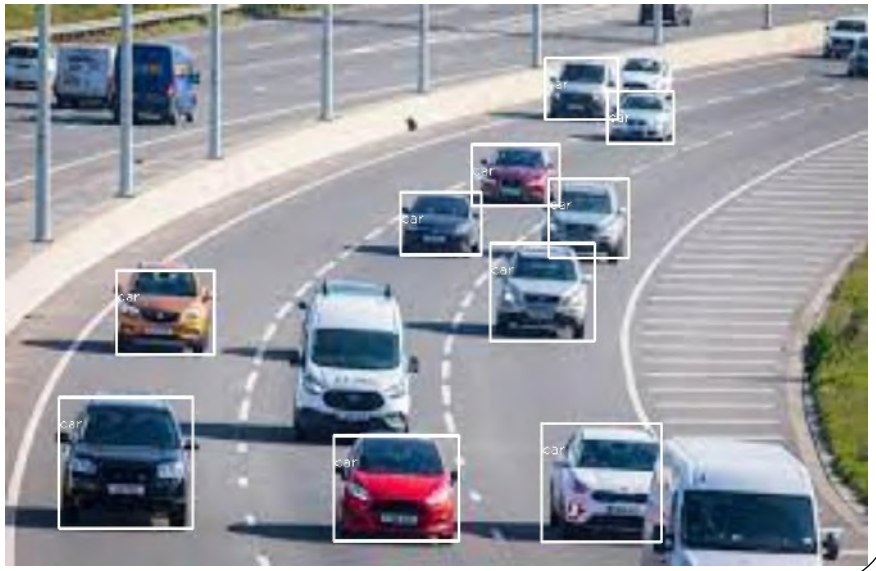
Python Output

```
 RESTART: D:\work\SNIST\major project 4-2\codes\yolo\yolo_updated_dynamic_4lane_ambulance.py
Lanes given Green ->  <-
Enter Initial fo
27
HIGH TRAFFIC
time = 6.0
count = 27.0
Enter Initial fo
1
LOW TRAFFIC
time = 2.0
count = 1.0
Enter Initial fo
ambulance
green for 2 beca
Enter Ambulance
ambulance
green for 2 beca
Enter Ambulance
0
NO TRAFFIC
green blinked at
Lanes given Gree
time = 0.0
count = -1.0
Enter Initial fo
27
HIGH TRAFFIC
time = 6.0
count = 27.0
red for all
green for 3
Lanes given Gree
Enter dynamic:4
```

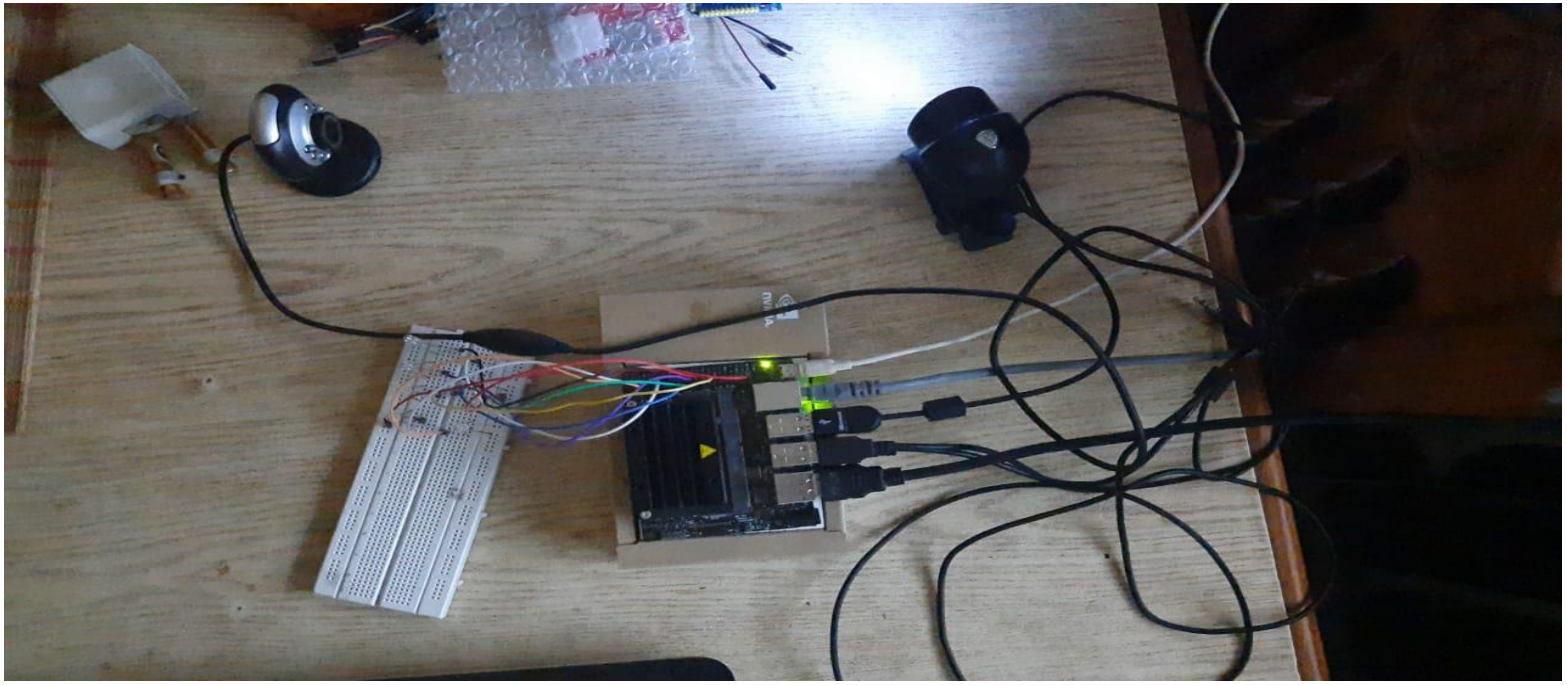

```
Python 2.7.16 (v2.7.16:413a49145e, Mar  4 2019, 01:30:55) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
 RESTART: D:\work\SNIST\major project 4-2\codes\yolo\yolo_updated_dynamic_4lane_ambulance.py
Lanes given Green ->  <-
Enter Initial fo
27
HIGH TRAFFIC
time = 6.0
count = 27.0
Enter Initial fo
1
LOW TRAFFIC
time = 2.0
count = 1.0
Enter Initial fo
ambulance
green for 2 beca
Enter Ambulance
ambulance
```

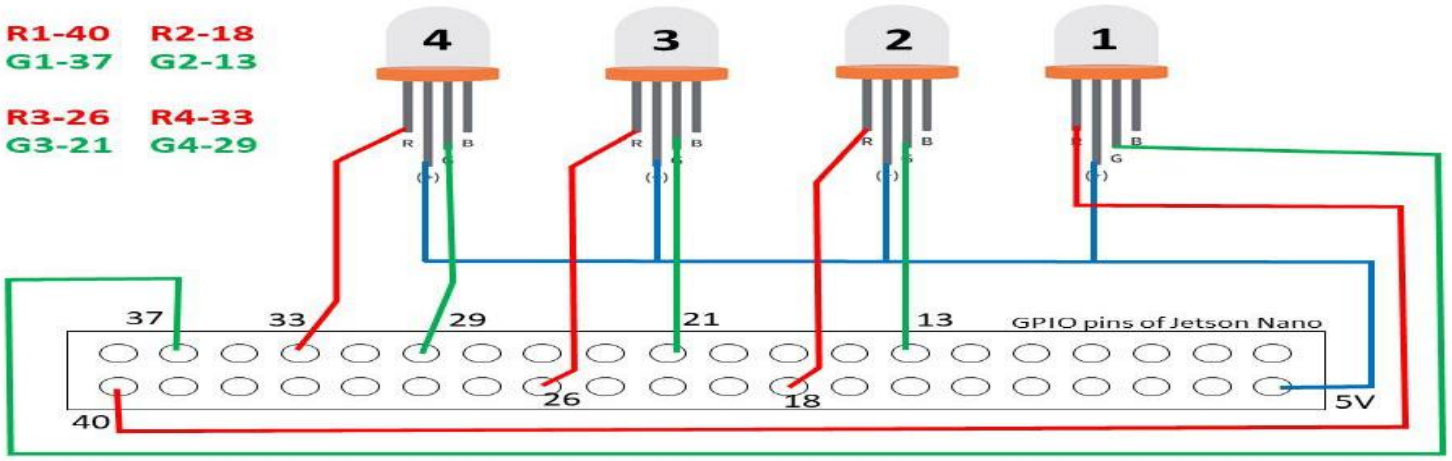
Output Images



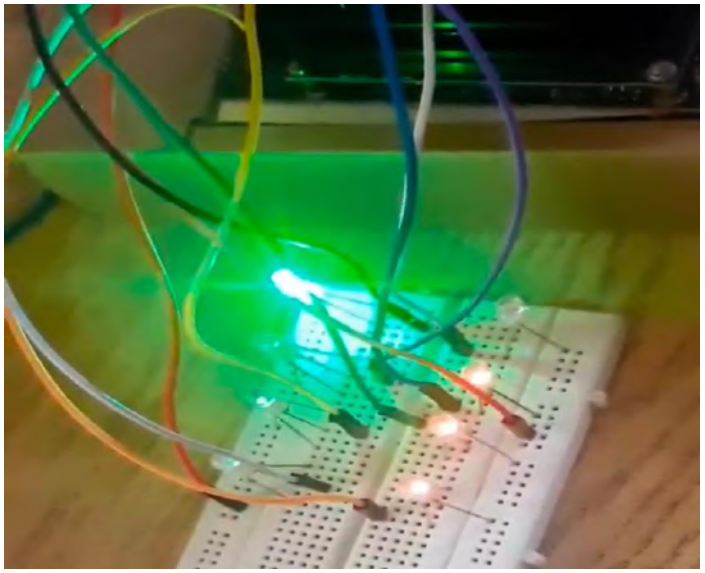


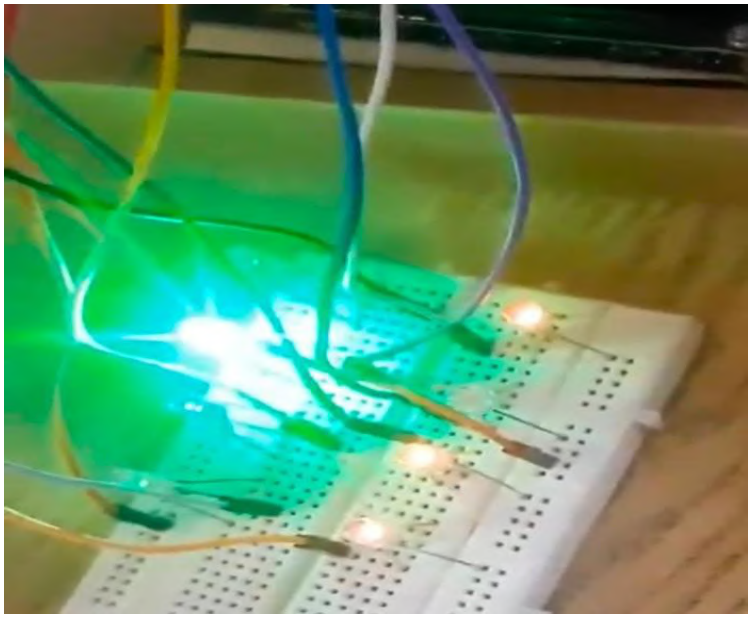
NVIDIA Circuit Connection

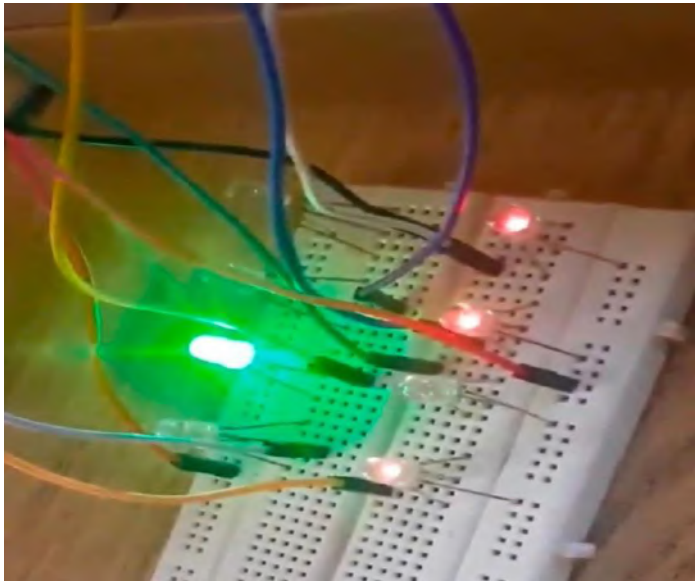


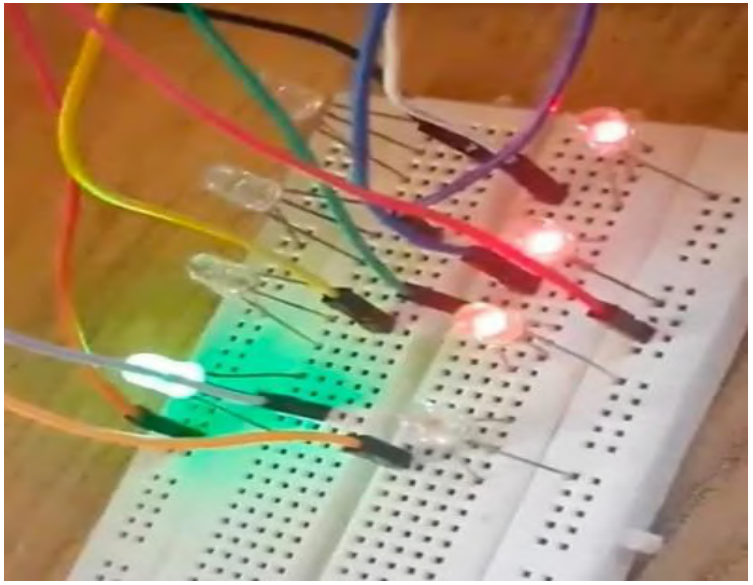


NVIDIA Traffic Signal Output









WORKING VIDEO LINK

- https://www.youtube.com/watch?v=EEvOt_BgOqk

APPLICATIONS

- The proposed method focuses on overcoming the traffic congestion scenarios experienced.
- In our project, we measure the traffic density so that each signal gets only required amount of time to clear the congestion. All this process is done at real time using image/ video processing by taking the input from cameras already fixed at each signal.
- We even try to spot an ambulance so as to prioritize it. For the ambulance detection, we use shape detection and color detection so as to spot the red cross symbol on the ambulance.
- With this solution, we can ensure that traffic congestion can be cleared in a fast and hassle free manner. This reduces the overall waiting time and results in a smoother traffic flow.

THANK YOU