TRAFFIC MANAGEMENT

Objective:

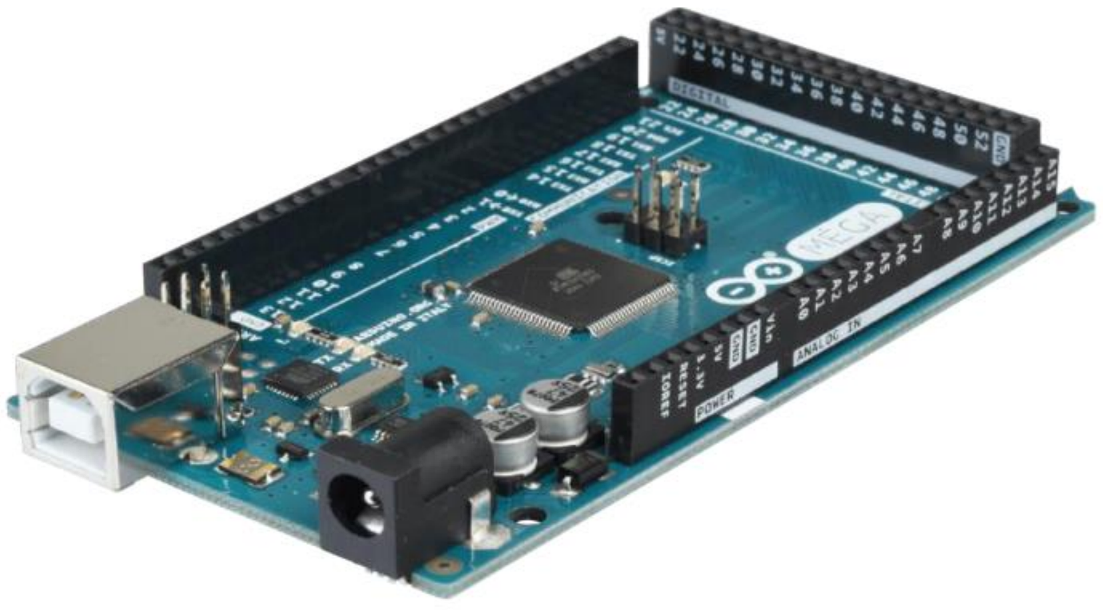
"Enhance traffic management systems through the strategic deployment of IoT technologies to achieve safer, more efficient, and sustainable urban transportation networks”.

- Real-time Traffic Monitoring: Implement IoT sensors and cameras to continuously monitor traffic flow, congestion, and incidents in real time.
- Data Collection and Analysis: Gather and analyze data from IoT devices to gain insights into traffic patterns, peak hours, and areas prone to congestion.
- Traffic Optimization: Develop algorithms and systems that use IoT data to optimize traffic signals, prioritize public transportation, and reduce gridlock.
- Predictive Maintenance: Employ IoT sensors to monitor the condition of road infrastructure, traffic lights, and signs to perform proactive maintenance and reduce downtime.

Traffic management is one of the biggest infrastructure hurdles faced by developing countries today. Developed countries and smart cities are already using IoT and to their advantage to minimize issues related to traffic. The culture of the car has been cultivated speedily among people in all types of nations. In most cities, it is common for people to prefer riding their own vehicles no matter how good or bad the public transportation is or considering how much time and money is it going to take for them to reach their destination.

HARDWARE REQUIREMENTS:

1. Microcontroller (Arduino Mega 2560): The Arduino Mega 2560 is a micro□controller board based on the Atmega 2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega 2560 board is compatible with most shields designed for the Uno and the former boards Duemilanove or Diecimila.



2. Microcontroller (Arduino Uno): The Arduino UNO is an open-source micro□controller board based on the Microchip ATmega328Pmicrocontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 Digital pins, 6 Analog pins, and programmable with the

Arduino IDE (Integrated Development Environment) via a type B USB cable.

3. **LEDs:** LEDs are used for the purpose of signaling according to the traffic condition.

4. **IR Sensor:** IR Sensor is used to count the vehicles on the road.

5. **Jumper Wires:** It is used to connect the components to each other.

SOFTWARE REQUIREMENTS:

1. **PYTHON:**
   We implement a python program for to import IoT devices record some activites using the required hardware analyze the result of data daily.

2. **HTML,CSS,JS:**
   Here HTML is used to develop a web interface app for all platform interfaces.

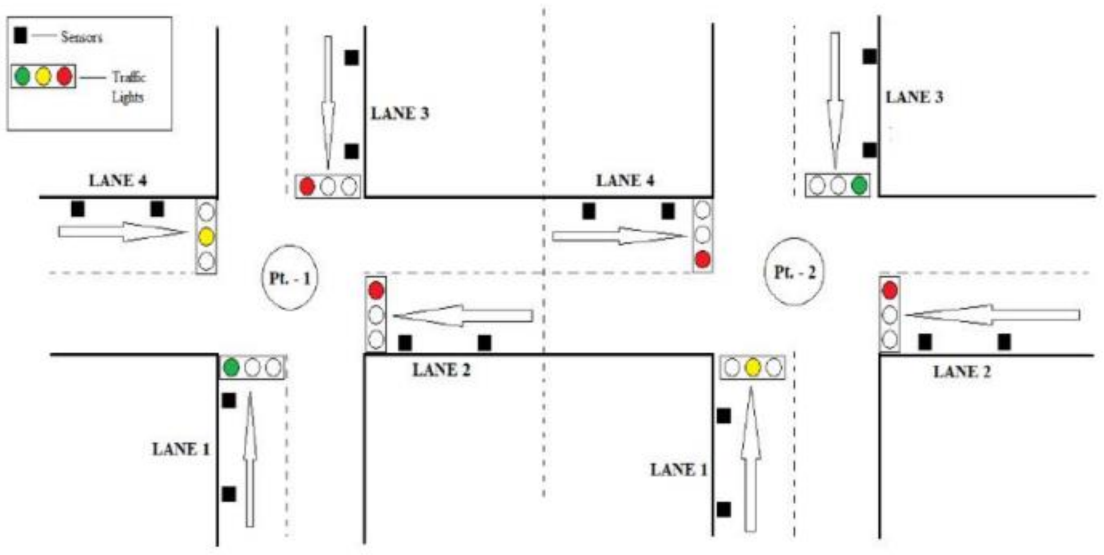
ADVANTAGES:

i) Minimizes number of accidents.
ii) Reduces fuel cost and saves time.
iii) Low budget.
iv) Easy implementation and maintenance.
v) Remotely controllable.
vi) Minimizes hassle and cost of commuting.

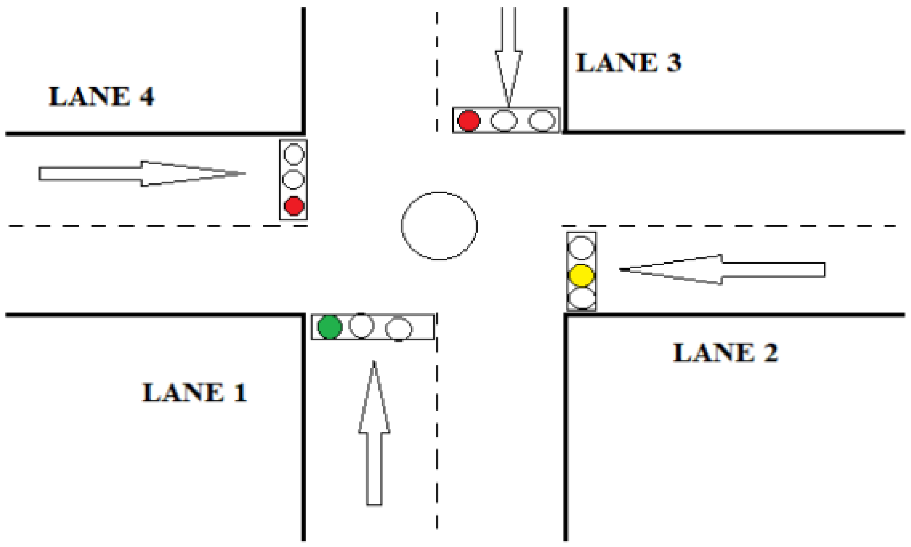
METHOD:

In this system, the traffic lights are LEDs and the car counting sensor is an ultrasonic sensor. Both blocks are connected to a Microcontroller using physical wires. The Microcontroller is the traffic light controller which receives the collected sensor data and manages the traffic lights by switching between green, yellow and red. The Microcontroller computes the number of cars in the street of the inter□section it is monitoring based on the distances measured by the ultrasonic sensor and the timing between those measurements. The Microcontroller then sends the number of cars every minute to the local server. This communication is done using the Microcontroller serial port. The local server exchanges the data received with the cloud server in order to better predict the changes in timings of the traffic light. This communication is done using Wi-Fi. More specifically, the cloud server uses an equation that takes the data received (number of cars) as input then determines the time interval of LEDs needed for a smooth traffic flow. This calculated time is then compared to the current actual time of the LEDs (this data is saved in a database on the cloud server). The server then comes up with a decision. If the current actual green time is less than the calculated time, the decision is to increase the green time, else to decrease the green time.

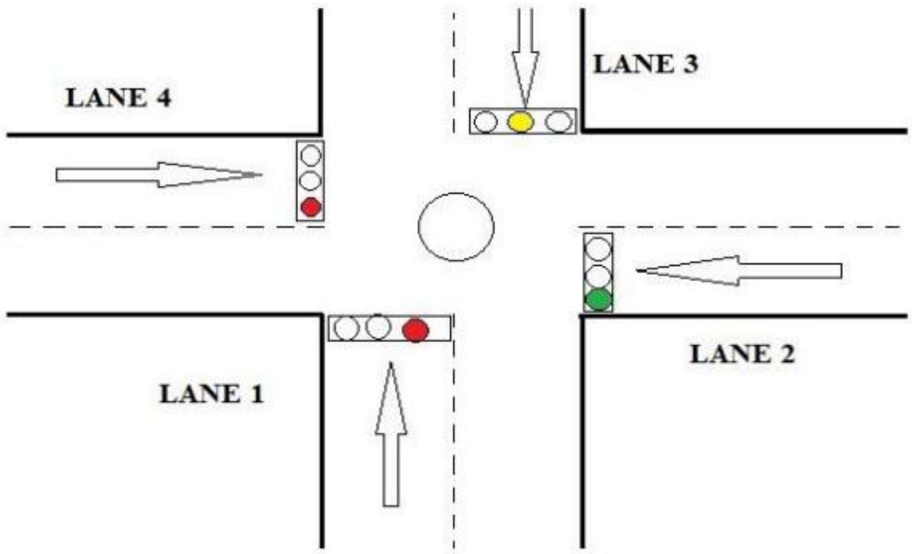
A View of Signals at Different Lanes:



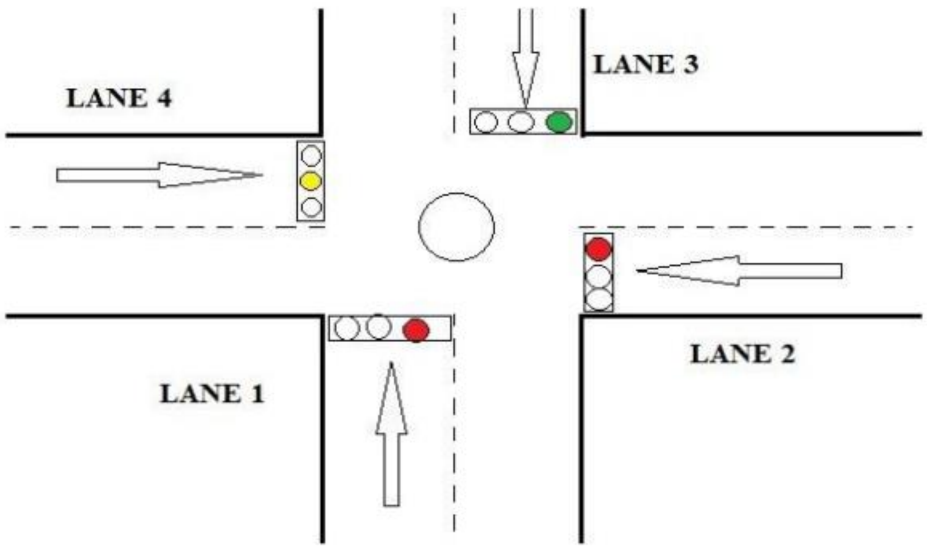
In the above figure, in Pt. - 1, LANE 1 is currently open with green signal and LANE 4 is ready with an yellow signal but LANE 2 and LANE 3 are blocked. In LANE 3, vehicle count is already greater than the threshold value, therefore the road coming to LANE 2 of Pt. - 1 is blocked in the Pt. - 2 itself. Thus re-routing them through another lanes. (Assuming that Pt. - 1 is the current intersection and Pt. - 2 is the previous intersection.)



In the above figure, Lane 1 is open with green signal and other lanes are closed with red signal.



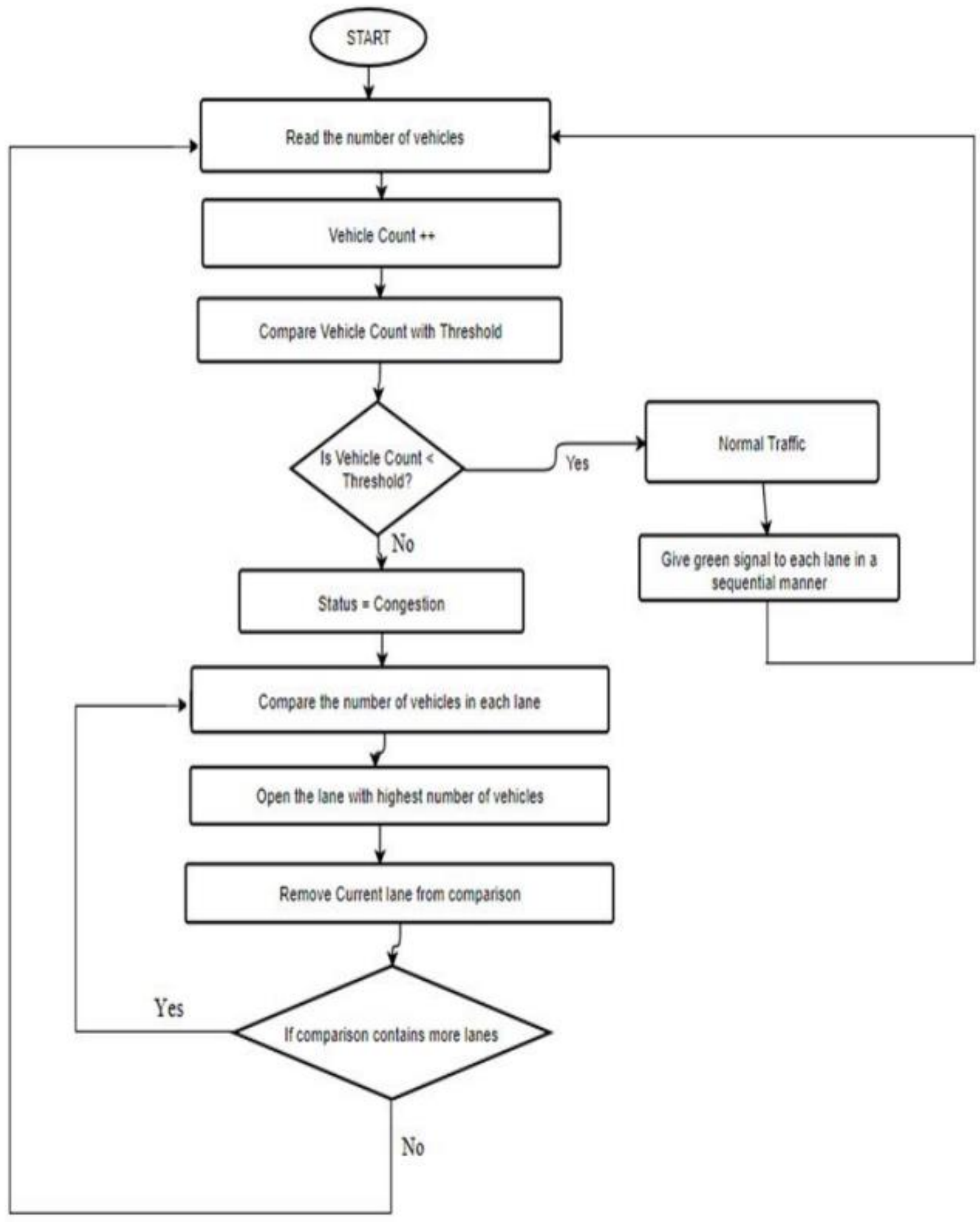
In the above figure, Lane 2 is open with green signal and other lanes are closed with red signal.



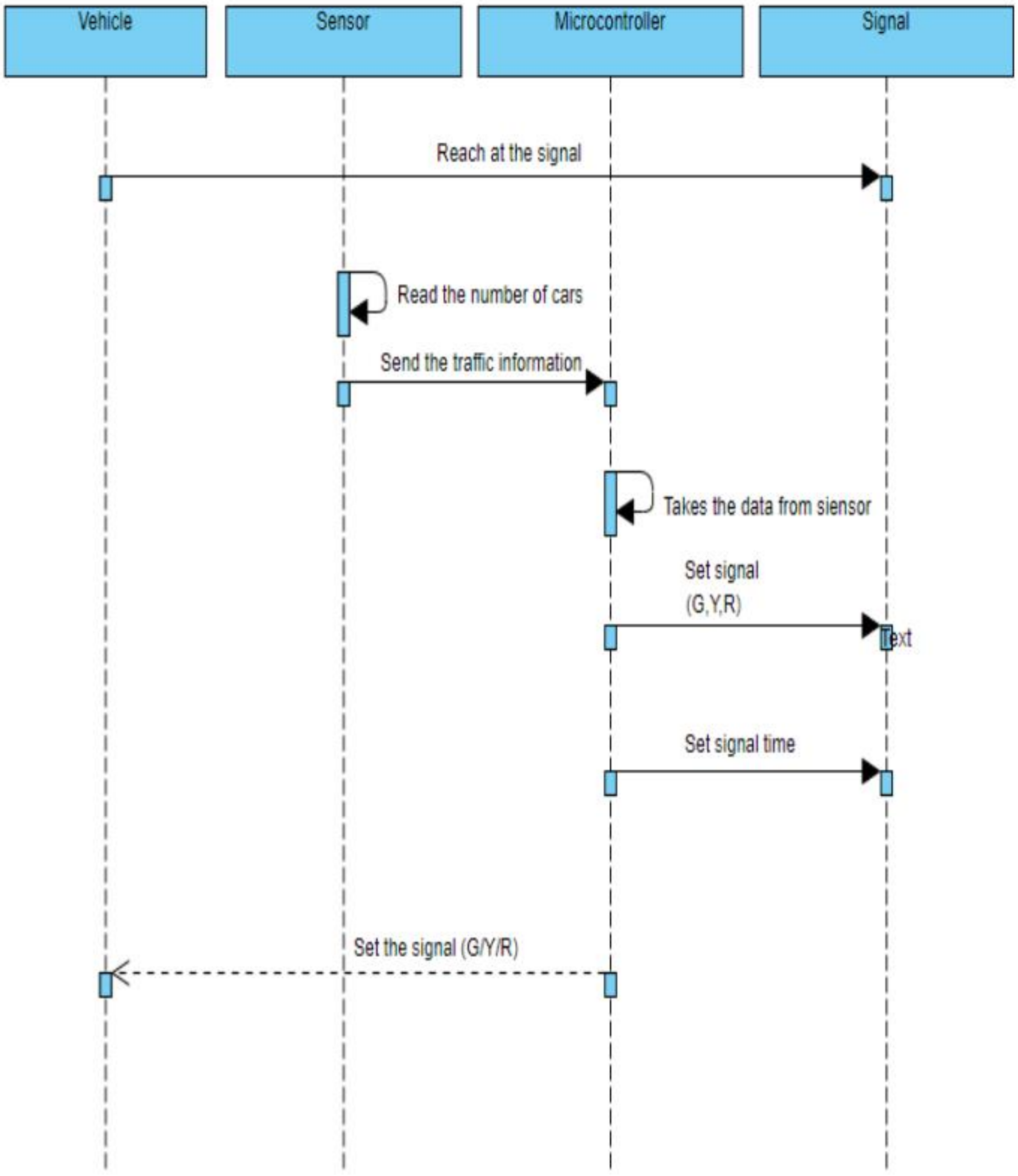
In the above figure, Lane 3 is open with green signal and other lanes are closed with red signal and after that Lane 4 will get the green signal automatically.

DIAGRAMS:

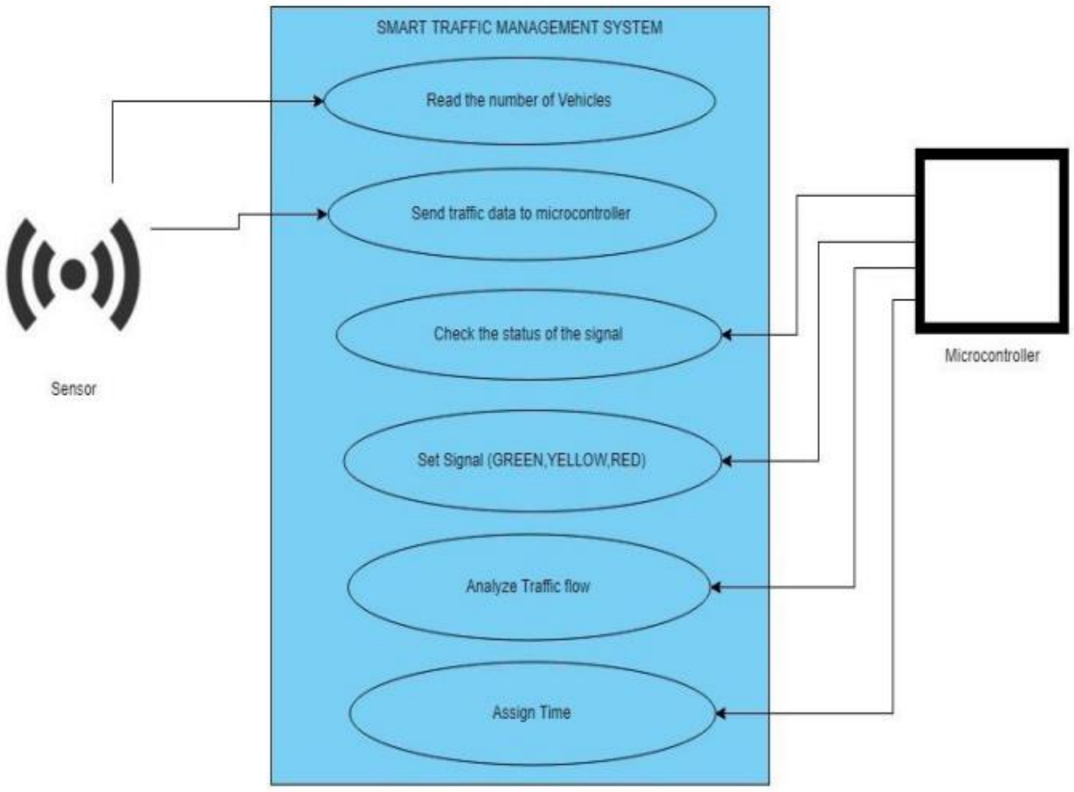
i) Flowchart


START
Read the number of vehicles
Vehicle Count ++
Compare Vehicle Count with Threshold
Is Vehicle Count < Threshold?
Yes
Normal Traffic
Give green signal to each lane in a sequential manner
No
Status = Congestion
Compare the number of vehicles in each lane
Open the lane with highest number of vehicles
Remove Current lane from comparison
Yes
If comparison contains more lanes
No

ii) Sequence diagram



iii) Use Case Diagram:


SMART TRAFFIC MANAGEMENT SYSTEM
Read the number of Vehicles
Send traffic data to microcontroller
Check the status of the signal
Set Signal (GREEN,YELLOW,RED)
Analyze Traffic flow
Assign Time
Sensor
Microcontroller

Vehicle Counting Algorithm:

Assuming the objects detected by the IR Sensors to be vehicles,

int counter = 0;

int hitObject = false;

int val ;

Step 1: Read value from sensor (val). Sensor gives output 0 if car is detected and 1 if no car is detected.

Step 2: If val == 0 hitObject = false then increment the counter and set hitObject = true.

else if val == 1 hitObject = true

then set hitObject = false.

Step 3: Go to step 1

Program for Vehicle Counting:

```
import numpy as np
import cv2
# Loads the data as a VideoCapture format, which is really just
# an image sequence.
image_sequence = 'day1.mp4'
cap = cv2.VideoCapture(image_sequence)
# Load our cascade classifier from cars3.xml
car_cascade = cv2.CascadeClassifier(r'/home/aman/opncv_tutorials/cars3.xml')
# Reduce frame number of tests.
number_of_frames_to_load = 500
for frame_id in xrange(number_of_frames_to_load):
    ret, image = cap.read()
    # Crop so that only the roads remain, eliminates the distraction.
    image=image[120:,:-20]
     #Use Cascade Classifier to detect cars, may have to tune the
     #parameters for less false positives.
    cars = car_cascade.detectMultiScale(image, 1.008, 5)
    for (x,y,w,h) in cars:
        cv2.rectangle(image,(x,y),(x+w,y+h),(255,0,0),2)
    print 'Processing %d : cars detected : [%s]' % (frame_id, len(cars))
    cv2.imshow('frame', image)
    cv2.waitKey(10)
cap.release()
cv2.destroyAllWindows()
def ir_sensor():
    val = 1 # Initialize to no car detected
    while True:
        # Step 1: Read value from sensor (val)
        print("Reading sensor...")
        if val == 0:
            print("Car detected!")
        elif val == 1:
            print("No car detected.")
        else:
            print("Invalid sensor value.")
            continue

        # Step 2: Process detected car
        if val == 0:
            print("Hit Object = ", hitObject)
            if not hitObject:
                counter += 1
                hitObject = True
            else:
                print("Hit counter increased. Counter: ", counter)
                hitObject = False

        # Step 3: Go to step 1
        val = 1 # Set back to no car detected for next loop iteration

ir_sensor()
```

Traffic Control Algorithm:

No. of sensors = 8 and are denoted by S1, S2, S3, S4, S5, S6, S7, S8

No. of cars in Lane 1 (N1) = S1 – S2

No. of cars in Lane 2 (N2) = S3 – S4

No. of cars in Lane 3 (N3) = S5 – S6

No. of cars in Lane 4 (N4) = S7 – S8

Li = (L1, L2, L3, L4), Ni = (N1, N2, N3, N4), Ti = (T1, T2, T3, T4)

```
Step 1: Start
Step 2: Sensors will read the no. of vehicles on each lane (i.e. L1, L2, L3, L4)
Step 3: if (Vehicle Count < Threshold)
Then status = Normal traffic. Turn on the green signal for all the lanes one after
another in a sequential manner (L1-L2-L3-L4). When signal is green for one lane,the
others will remain red.
Step 4: else status = congestion.
Step 5: COMPARE (N1, N2 , N3, N4), Select the highest of the four (say Ni),turn
on green signal for that lane (say Li) for time (Ti). When time Ti ends, turn on the
red signal.
Step 6: COMPARE (N2, N3, N4), Select the highest of the three (say Ni), turn on
green signal for that lane (say Li) for time (Ti). When time Ti ends, turn on the
red signal.
Step 7: COMPARE (N3, N4), Select the highest of the two (say Ni), turn on green
signal for that lane (say Li) for time (Ti). When time Ti ends, turn on the red
```

signal.

Step 8: The last remaining lane automatically gets selected and it is given the green

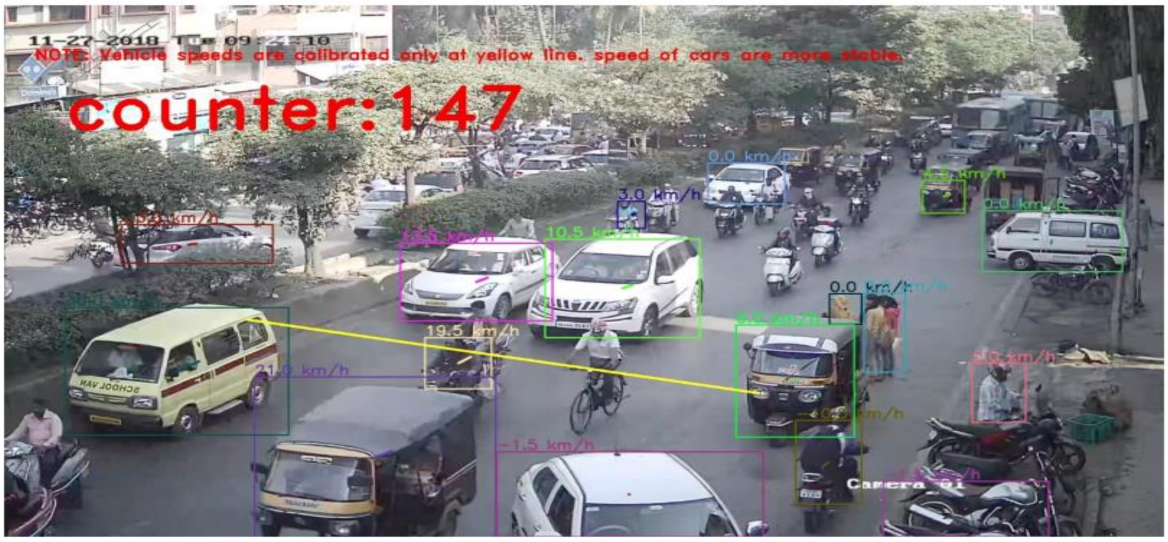
signal for time Ti.

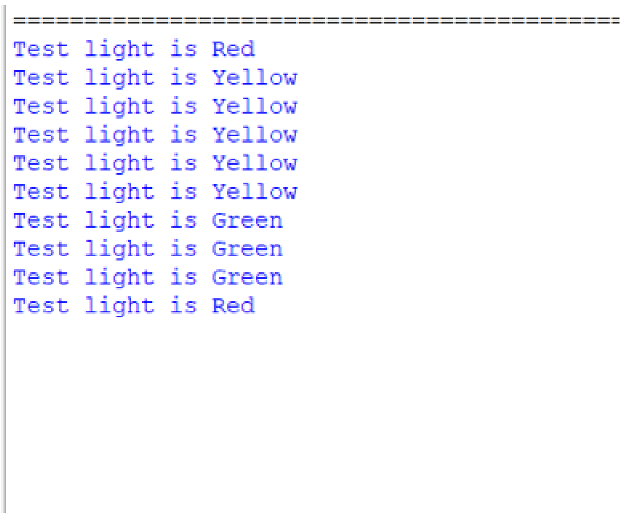
Step 9: Jump to Step 3.

Program for Traffic Control:

```python
import numpy as np
import cv2

cap = cv2.VideoCapture('traffic.mp4')

kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3))
fgbg = cv2.bgsegm.createBackgroundSubtractorGMG()

while(1):
    ret, frame = cap.read()

    fgmask = fgbg.apply(frame)
    fgmask = cv2.morphologyEx(fgmask, cv2.MORPH_OPEN, kernel)

    cv2.imshow('frame',fgmask)
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

cap.release()
cv2.destroyAllWindows()

def green_light(n1, n2, n3, n4):
    while True:
        # case when vehicle count is less than threshold
        if n1 < 10 and n2 < 10 and n3 < 10 and n4 < 10:
            for lane in [1, 2, 3, 4]:
                green_signal(lane)
                time.sleep(5) # assuming 5 seconds per lane
        # case when traffic is congested
        else:
            lane_count = sorted([n1, n2, n3, n4], reverse=True)
            lane_to_signal = lane_count[0]
            green_signal(lane_to_signal)
            time.sleep(5) # assuming 5 seconds
            red_signal(lane_to_signal)
            if lane_count[1] != lane_count[0]:
                lane_to_signal = lane_count[1]
                green_signal(lane_to_signal)
                time.sleep(5) # assuming 5 seconds
                red_signal(lane_to_signal)
            if lane_count[2] != lane_count[1] and lane_count[2] != lane_count[0]:
                lane_to_signal = lane_count[2]
                green_signal(lane_to_signal)
                time.sleep(5) # assuming 5 seconds
                red_signal(lane_to_signal)
            if lane_count[3] != lane_count[2] and lane_count[3] != lane_count[1] and
lane_count[3] != lane_count[0]:
                lane_to_signal = lane_count[3]
                green_signal(lane_to_signal)
                time.sleep(5) # assuming 5 seconds
                red_signal(lane_to_signal)
```

SCREENSHOT:


11-27-2018
NOTE: Vehicle speeds are calibrated only at yellow line. speed of cars are more stable.
counter:147
0.0 km/h
3.0 km/h
10.5 km/h
19.5 km/h
21.0 km/h
-1.5 km/h



```
============================================
Test light is Red
Test light is Yellow
Test light is Yellow
Test light is Yellow
Test light is Yellow
Test light is Yellow
Test light is Green
Test light is Green
Test light is Green
Test light is Red
```

CODE FOR APP DEVELOPMENT :

HTML

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Traffic Management</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
        <h1>Traffic Management System</h1>
        <div class="light" id="red"></div>
        <div class="light" id="yellow"></div>
        <div class="light" id="green"></div>
    </div>

    <script src="scripts.js"></script>
</body>
</html>
```

CSS

```css
body {
    background-color: #222;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
}

.container {
    display: flex;
    flex-direction: column;
    align-items: center;
}
```

```css
h1 {
    color: #fff;
    margin-bottom: 50px;
}

.light {
    width: 100px;
    height: 100px;
    margin: 10px;
    border-radius: 50%;
    opacity: 0.2;
    transition: opacity 0.3s;
}

#red {
    background-color: red;
}

#yellow {
    background-color: yellow;
}

#green {
    background-color: green;
}
```

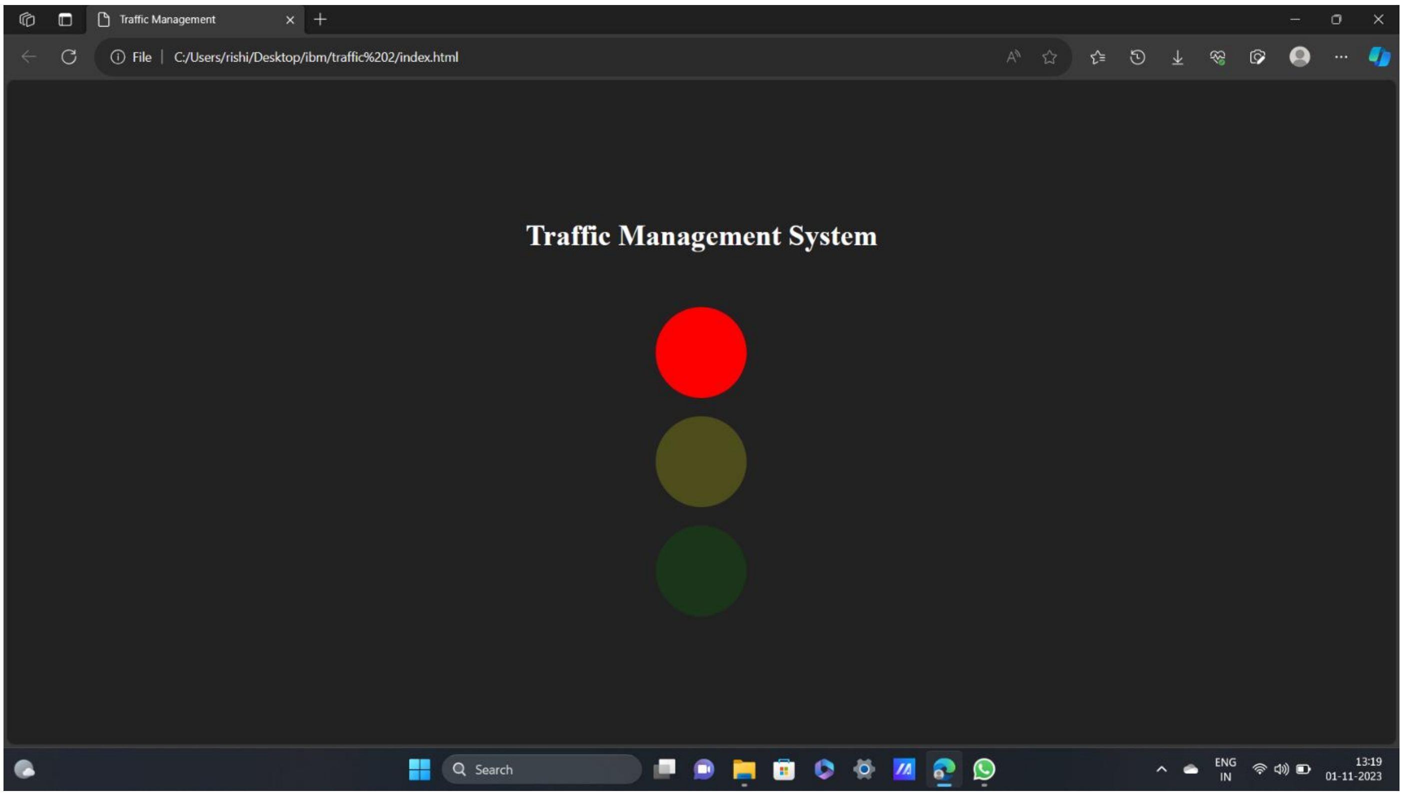
JS

```javascript
let redLight = document.getElementById('red');
let yellowLight = document.getElementById('yellow');
let greenLight = document.getElementById('green');

function changeLight() {
    greenLight.style.opacity = 0.2;
    yellowLight.style.opacity = 0.2;
    redLight.style.opacity = 1;
    setTimeout(() => {
        redLight.style.opacity = 0.2;
        yellowLight.style.opacity = 1;
    }, 3000);
    setTimeout(() => {
        yellowLight.style.opacity = 0.2;
        greenLight.style.opacity = 1;
    }, 6000);
}

changeLight();
setInterval(changeLight, 9000);
```

SERVER CODE:

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html')

if __name__ == '__main__':
    app.run(debug=True)
```

SCREENSHOT:



Traffic Management System

CONCLUSION:

i) The proposed system helps in better time based monitoring and thus has certain advantages over the existing system like minimizing number of accidents, reducing fuel cost and is remotely controllable etc.

ii) The proposed system is designed in such a way that it will be able to control the traffic congestion as well as track the number of vehicles. The administrator of the system can access local server in order to maintain the system.