TRAFFICMANAGEMENT USING IoT

# NAME:  AVIKA M

COLLEGECODE:9530

NAN MUDHALVAN ID:  au953021106011

**PHASE-4 PROJECT SUBMISSION**

PROJECT TITLE:  TRAFFIC MANAGEMENT SYSTEM

PHASE4:  DEVELOPEMENT PART 2

# INTRODUCTION:

PROBLEM STATEMENT:

Traffic congestion consists of incremental delay, consumption, pollution emission and stress that result from interference vehicles in traffic stream, particularly as traffic volumes approaches a road’s capacity.

DESCRIPTION:

Traffic management concerns the control, planning, and purchasing of transport services needed to physically move road vehicles.

COMPONENTS REQUIRED:

* ARDUINO UNO
* TRAFFIC SENSOR
* JUMPER WIRES
* BREAD BOARD
* LED

### SOFTWARE REQUIRED:

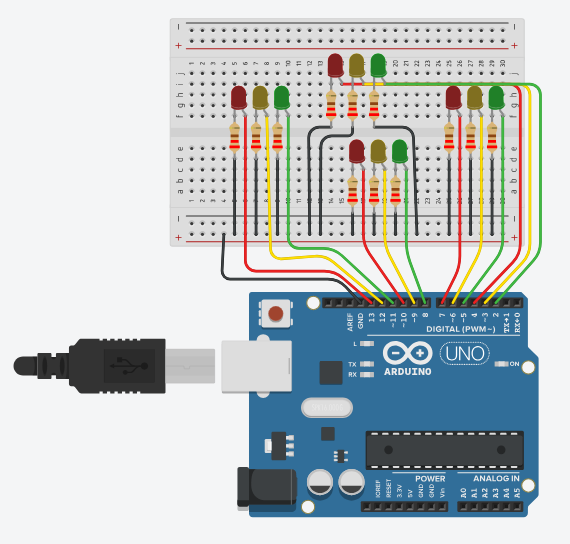
* Arduino IDE:

Used to connect Arduino UNO and upload the file into them for further processing.

* TINKERCAD:

Used for creating digital circuit and to simulate that.

# **CIRCUIT DIAGRAM:**



# SOURCE CODE TO SIMULATE ARDUINO:

int red\_1=13;  
int orange\_1=12;  
int green\_1=11;  
int red\_2=10;  
int orange\_2=9;  
int green\_2=8;  
int red\_3=7;  
int orange\_3=6;  
int green\_3=5;  
int red\_4=4;  
int orange\_4=3;  
int green\_4=2;  
  
void direction\_1\_green(void)  
{  
    digitalWrite(red\_1,LOW);  
    digitalWrite(orange\_1,LOW);  
    digitalWrite(green\_1,HIGH);  
    digitalWrite(red\_2,HIGH);  
    digitalWrite(orange\_2,LOW);  
    digitalWrite(green\_2,LOW);  
    digitalWrite(red\_3,HIGH);  
    digitalWrite(orange\_3,LOW);  
    digitalWrite(green\_3,LOW);  
    digitalWrite(red\_4,HIGH);  
    digitalWrite(orange\_4,LOW);  
    digitalWrite(green\_4,LOW); }  
  
void direction\_2\_orange(void)  
{  
    digitalWrite(red\_1,HIGH);  
    digitalWrite(orange\_1,LOW);  
    digitalWrite(green\_1,LOW);  
    digitalWrite(red\_2,LOW);  
    digitalWrite(orange\_2,HIGH);  
    digitalWrite(green\_2,LOW);  
    digitalWrite(red\_3,HIGH);  
    digitalWrite(orange\_3,LOW);  
    digitalWrite(green\_3,LOW);  
    digitalWrite(red\_4,HIGH);  
    digitalWrite(orange\_4,LOW);  
    digitalWrite(green\_4,LOW);  
     
}  
  
void direction\_2\_green(void)  
{  
    digitalWrite(red\_1,HIGH);  
    digitalWrite(orange\_1,LOW);  
    digitalWrite(green\_1,LOW);  
    digitalWrite(red\_2,LOW);  
    digitalWrite(orange\_2,LOW);  
    digital Write(green\_2,HIGH);  
    digitalWrite(red\_3,HIGH);  
    digitalWrite(orange\_3,LOW);  
    digitalWrite(green\_3,LOW);  
    digitalWrite(red\_4,HIGH);  
    digitalWrite(orange\_4,LOW);  
    digitalWrite(green\_4,LOW);  
     
}  
  
void direction\_3\_orange(void)  
  
{  
    digitalWrite(red\_1,HIGH);  
    digitalWrite(orange\_1,LOW);  
    digitalWrite(green\_1,LOW);  
    digitalWrite(red\_2,HIGH);  
    digitalWrite(orange\_2,LOW);  
    digitalWrite(green\_2,LOW);  
    digitalWrite(red\_3,LOW);  
    digitalWrite(orange\_3,HIGH);  
    digitalWrite(green\_3,LOW);  
    digitalWrite(red\_4,HIGH);  
    digitalWrite(orange\_4,LOW);  
    digitalWrite(green\_4,LOW);  
    }  
  
void direction\_3\_green(void)  
{  
    digitalWrite(red\_1,HIGH);  
    digitalWrite(orange\_1,LOW);  
    digitalWrite(green\_1,LOW);  
    digitalWrite(red\_2,HIGH);  
    digitalWrite(orange\_2,LOW);  
    digitalWrite(green\_2,LOW);  
    digitalWrite(red\_3,LOW);  
    digitalWrite(orange\_3,LOW);  
    digitalWrite(green\_3,HIGH);  
    digitalWrite(red\_4,HIGH);  
    digitalWrite(orange\_4,LOW);  
    digitalWrite(green\_4,LOW);  
    }  
    void direction\_4\_orange(void)  
{  
    digitalWrite(red\_1,HIGH);  
    digitalWrite(orange\_1,LOW);  
    digitalWrite(green\_1,LOW);  
    digitalWrite(red\_2,HIGH);  
    digitalWrite(orange\_2,LOW);  
    digitalWrite(green\_2,LOW);  
    digitalWrite(red\_3,HIGH);  
    digitalWrite(orange\_3,LOW);  
    digitalWrite(green\_3,LOW);  
    digitalWrite(red\_4,LOW);  
    digitalWrite(orange\_4,HIGH);  
    digitalWrite(green\_4,LOW);  
    }  
    void direction\_4\_green(void)  
    //green LED of direction 4 will turn ON  
{  
     digitalWrite(red\_1,HIGH);  
     digitalWrite(orange\_1,LOW);  
     digitalWrite(green\_1,LOW);  
     digitalWrite(red\_2,HIGH);  
     digitalWrite(orange\_2,LOW);  
     digitalWrite(green\_2,LOW);  
     digitalWrite(red\_3,HIGH);  
     digitalWrite(orange\_3,LOW);  
     digitalWrite(green\_3,LOW);  
     digitalWrite(red\_4,LOW);  
     digitalWrite(orange\_4,LOW);  
     digitalWrite(green\_4,HIGH);  
     }  
     void direction\_1\_orange(void)  
     //orange LED of direction 1 will turn ON  
     {  
         
         digitalWrite(red\_1,LOW);  
          
        digitalWrite(orange\_1,HIGH);  
        digitalWrite(green\_1,LOW);  
        digitalWrite(red\_2,HIGH);  
        digitalWrite(orange\_2,LOW);  
        digitalWrite(green\_2,LOW);  
        digitalWrite(red\_3,HIGH);  
        digitalWrite(orange\_3,LOW);  
        digitalWrite(green\_3,LOW);  
        digitalWrite(red\_4,HIGH);  
        digitalWrite(orange\_4,LOW);  
        digitalWrite(green\_4,LOW); }  
        void setup()  
        {  
            // Declaring all the LED's as output  
  
for(int I=2;i<=13;i++)  
pinMode(I, OUTPUT);  
  
} void loop()  
//In the loop function, we controlled the signal one // by one to control the flow of traffic.  
{  
    direction\_1\_green();  
    delay([5000](tel:5000));  
    direction\_2\_orange(); delay([3000](tel:3000));  
    direction\_2\_green(); delay([5000](tel:5000));  
    direction\_3\_orange(); delay([3000](tel:3000));  
    direction\_3\_green(); delay([5000](tel:5000));  
    direction\_4\_orange(); delay([3000](tel:3000));  
    direction\_4\_green(); delay([5000](tel:5000));  
    direction\_1\_orange(); delay([3000](tel:3000)); }

# UPLOAD THE CODE TO YOUR ARDUINO UNO:

* Open the ARDUINO IDE on your computer.
* connect your Arduino UNO to your computer via USB.
* Select the correct board and plot under the “tools “menu.
* copy and paste the above code into the Arduino IDE.
* click the “upload” button to upload the code to your Arduino Uno.
* simulate the code to get the output.

# TRAFFIC SYSTEM:

# 

# SOURCE  CODE IN PYTHON:

import time  
import random  
  
class Vehicle:  
    def \_\_init\_\_(self, vehicle\_id, speed):  
        self.vehicle\_id = vehicle\_id  
        [self.speed](http://self.speed) = speed  
  
class TrafficLight:  
    def \_\_init\_\_(self):  
        [self.state](http://self.state) = 'red'  
  
    def switch\_state(self):  
        [self. State](http://self.state) = 'green' if [self.state](http://self.state) == 'red' else 'red'  
  
class TrafficManager:  
    def \_\_init\_\_(self, num\_vehicles, num\_iterations):  
        self. vehicles = [Vehicle(i, [random.randint](http://random.randint)(30, 70)) for i in range(num\_vehicles)]  
        self.traffic\_light = Traffic Light()  
        self. num\_iterations = num\_iterations  
  
    def run\_simulation(self):  
        for iteration in range(self. num\_iterations):  
            print(f"Iteration {iteration + 1}: Traffic Light is {[self.traffic\_light.state](http://self.traffic_light.state)}")  
            for vehicle in [self. Vehicles](http://self.vehicles):  
                if [self.traffic\_light.state](http://self.traffic_light.state) == 'green':  
                    distance = [vehicle. speed](http://vehicle.speed) \* 2  
                    print(f"Vehicle {vehicle.vehicle\_id} moves {distance} meters.")  
                else:  
                    print(f"Vehicle {vehicle.vehicle\_id} stops at the red light.")  
            self.traffic\_light.switch\_state()  
            [time. Sleep](http://time.sleep)(1)  
  
def main():  
    num\_vehicles = 5  
    num\_iterations = 5  
    traffic\_manager = TrafficManager(num\_vehicles, num\_iterations)  
    traffic\_manager.run\_simulation()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
    main()

**OUTPUT:**

Vehicle 0 stops at the red light.  
Vehicle 1 stops at the red light.  
Vehicle 2 stops at the red light.  
Vehicle 3 stops at the red light.  
Vehicle 4 stops at the red light.  
  
Iteration 2: Traffic Light is green  
Vehicle 0 moves 80 meters.  
Vehicle 1 moves 50 meters.  
Vehicle 2 moves 70 meters.  
Vehicle 3 moves 40 meters.  
Vehicle 4 moves 60 meters.  
  
Iteration 3: Traffic Light is red  
Vehicle 0 stops at the red light.  
Vehicle 1 stops at the red light.  
Vehicle 2 stops at the red light.  
Vehicle 3 stops at the red light.  
Vehicle 4 stops at the red light.  
  
Iteration 4: Traffic Light is green  
Vehicle 0 moves 60 meters.  
Vehicle 1 moves 40 meters.  
Vehicle 2 moves 80 meters.  
Vehicle 3 moves 70 meters.  
Vehicle 4 moves 50 meters.  
  
Iteration 5: Traffic Light is red  
Vehicle 0 stops at the red light.  
Vehicle 1 stops at the red light.  
Vehicle 2 stops at the red light.  
Vehicle 3 stops at the red light.  
Vehicle 4 stops at the red light.

**EXPLANATION OF OUTPUT**:

 This code simulates a traffic system with 5 vehicles and 5 iterations. Each iteration represents a change in the traffic light state. When the traffic light is green, all vehicles move twice their speed. When the traffic light is red, all vehicles stop at the intersection. The simulation is run for 5 iterations, displaying the traffic light state and vehicle movement in each iteration.</s>

Please note that the actual output may vary due to the random nature of vehicle speeds and the traffic light switching time.</s> </s> Please also note that the output of the simulation includes a delay of 1 second after each iteration due to the time. sleep(1) function.

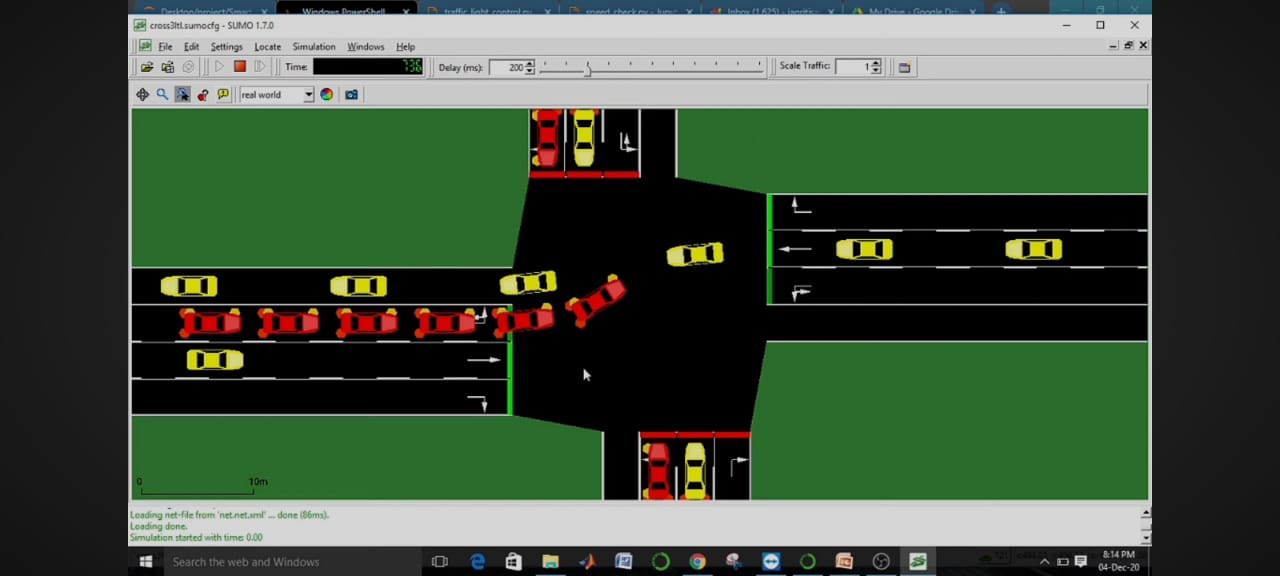


Figure:  sumo simulation

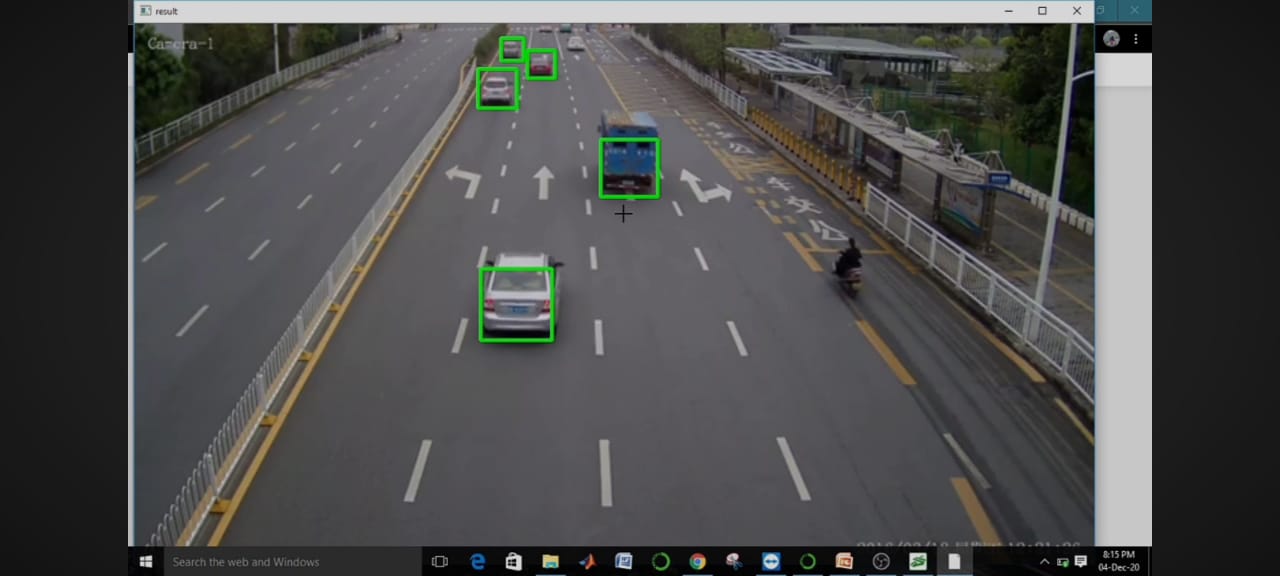
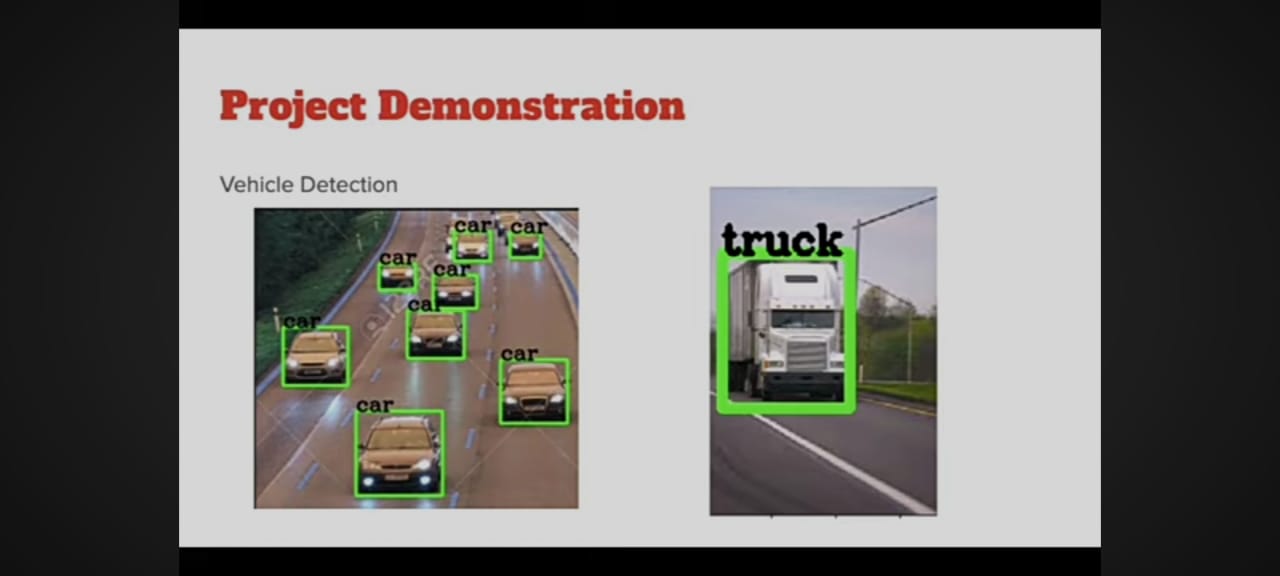


Figure:   Real time vehicle detection



**SOURCE CODE IN JAVASCRIPT:**

class Vehicle {

constructor(vehicle\_id, speed) {

this.vehicle\_id = vehicle\_id;

this. speed = speed;

}

}

class TrafficLight {

constructor() {

this. state = 'red';

}

switch\_state() {

this. state = this. state === 'red' ? 'green' : 'red';

}

}

class TrafficManager {

constructor(num\_vehicles, num\_iterations) {

this. vehicles = Array. from({ length: num\_vehicles }, (\_, i) => new Vehicle(i, Math. Floor(Math. Random() \* (70 - 30 + 1)) + 30));

this.traffic\_light = new TrafficLight();

this.num\_iterations = num\_iterations;

}

run\_simulation() {

for (let iteration = 0; iteration < this.num\_iterations; iteration++) {

console.log(Iteration ${iteration + 1}: Traffic Light is ${this.traffic\_light.state});

this.vehicles.forEach(vehicle => {

if (this.traffic\_light.state === 'green') {

const distance = vehicle.speed \* 2;

console.log(Vehicle ${vehicle.vehicle\_id} moves ${distance} meters.);

} else {

console.log(Vehicle ${vehicle.vehicle\_id} stops at the red light.);

}

});

this.traffic\_light.switch\_state();

sleep(1000); // JavaScript sleep function

}

}

}

function sleep(ms) {

return new Promise(resolve => set Timeout(resolve, ms));

}

function main() {

const num\_vehicles = 5;

const num\_iterations = 5;

const traffic\_manager = new TrafficManager(num\_vehicles, num\_iterations);

traffic\_manager.run\_simulation();

}

main();

**OUTPUT:**

Iteration 1: Traffic Light is red

Vehicle 0 stops at the red light.

Vehicle 1 stops at the red light.

Vehicle 2 stops at the red light.

Vehicle 3 stops at the red light.

Vehicle 4 stops at the red light.

Iteration 2: Traffic Light is green

Vehicle 0 moves 64 meters.

Vehicle 1 moves 138 meters.

Vehicle 2 moves 128 meters.

Vehicle 3 moves 76 meters.

Vehicle 4 moves 60 meters.

Iteration 3: Traffic Light is red

Vehicle 0 stops at the red light.

Vehicle 1 stops at the red light.

Vehicle 2 stops at the red light.

Vehicle 3 stops at the red light.

Vehicle 4 stops at the red light.

Iteration 4: Traffic Light is green

Vehicle 0 moves 64 meters.

Vehicle 1 moves 138 meters.

Vehicle 2 moves 128 meters.

Vehicle 3 moves 76 meters.

Vehicle 4 moves 60 meters.

Iteration 5: Traffic Light is red

Vehicle 0 stops at the red light.

Vehicle 1 stops at the red light.

Vehicle 2 stops at the red light.

Vehicle 3 stops at the red light.

Vehicle 4 stops at the red light.

**CONCLUSION:**

To summarize, the project involves the use of IoT devices such as traffic cameras, sensors, and traffic lights, along with a centralized platform for data collection, analysis, and visualization. The system aims to optimize traffic flow, enhance road safety, and promote environmental sustainability through real-time monitoring and predictive algorithms. By utilizing advanced data analytics and visualization techniques, the project facilitates informed decision-making for traffic authorities and enhances the overall efficiency of the transportation system.