PROJECT TITLE : TRAFFIC MANAGEMENT SYSTEM.

PROBLEM:

Congestion:

Urban areas frequently suffer from traffic congestion, leading to delays, increased fuel consumption, and air pollution.

DESIGN COMPONENT:

Hardware components:

1.Arduino Board

2.Infrared Sensors

3.Radar Sensors

4.Wi-Fi Module

5.LoRa Module

6.GSM Module

7.LED Displays

8.Traffic Signal Controllers

9.Power Supply

10.Casing and Mounting Hardware

Software components:

1.Arduino IDE

2.Arduino Sketch

3.Central Control System

4.Traffic Optimization Algorithms.

WORKING PRINCIPLE:

1.Data Collection:Sensors such as ultrasonic sensors, infrared sensors, and radar sensors are placed at key locations, like intersections or congested areas.These sensors detect vehicles, count their numbers, and gather information on their speed and presence.

2.Data Processing (Arduino):Arduino boards at the sensor locations collect data from the sensors.The Arduino boards process this data, performing tasks like counting vehicles, calculating average speeds, and detecting congestion.

3.Data Transmission (IoT Communication):Data processed by Arduino boards is transmitted to a central control system using an IoT communication module (e.g., GSM, Wi-Fi, or LoRa).The central control system can be located at a traffic management center or cloud-based server.

4.Central Control System:The central control system receives data from various sensor locations.It analyzes this data to assess traffic conditions and congestion levels.

5.Traffic Optimization Algorithms:Traffic optimization algorithms run on the central system to make decisions.Algorithms assess the real-time traffic data and adjust traffic signals or route traffic to reduce congestion.

6.User Alerts and Notifications:The central control system generates alerts and notifications for drivers, authorities, or traffic management personnel.Messages can be displayed on LED signs or sent through mobile apps to inform drivers of traffic conditions.

7.Traffic Signal Control:Arduino boards at traffic signal locations receive instructions from the central system.They adjust traffic signal timings based on real-time data to optimize traffic flow.

8.Integration with Public Transportation:The system integrates with public transportation services to provide real-time information about bus or train schedules.This encourages the use of public transit during congestion.

9.Traffic Enforcement Integration:Integration with traffic enforcement systems allows for automated enforcement of traffic rules, such as speeding violations, using cameras or other enforcement methods.

10.Environmental Monitoring:Environmental sensors continuously monitor air quality and emissions data.This data can be used to raise awareness of environmental impacts due to traffic congestion.

11.Data Security and Privacy:Security measures are in place to protect data and privacy, ensuring that sensitive information is not compromised.

12.Community Engagement:Display systems are set up at key locations to provide real-time traffic information to the community, helping drivers make informed decisions.

CIRCUIT EXPLANATION:

Circuit Connection:

1.Ultrasonic Sensor:

Connect the VCC (5V) pin of the ultrasonic sensor to the 5V pin on the Arduino.Connect the GND (ground) pin of the ultrasonic sensor to the GND pin on the Arduino.Connect the Echo pin of the ultrasonic sensor to a digital pin on the Arduino .Connect the Trigger pin of the ultrasonic sensor to another digital pin on the Arduino .

2.LED Lights:

Connect two LED lights to digital pins on the Arduino .Connect the anode (longer lead) of each LED to a digital pin and the cathode (shorter lead) to a current-limiting resistor (around 220-330 ohms) and then to the GND pin of the Arduino.

3.Power Supply:

Connect the Arduino to a power supply using a USB cable or an appropriate power source.

Circuit Operation:

1.The ultrasonic sensor emits ultrasonic waves and measures the time it takes for the waves to bounce back after hitting an object (a vehicle).

2.The Arduino reads the data from the ultrasonic sensor through the Echo and Trigger pins. It calculates the distance of the object (vehicle) based on the time taken.

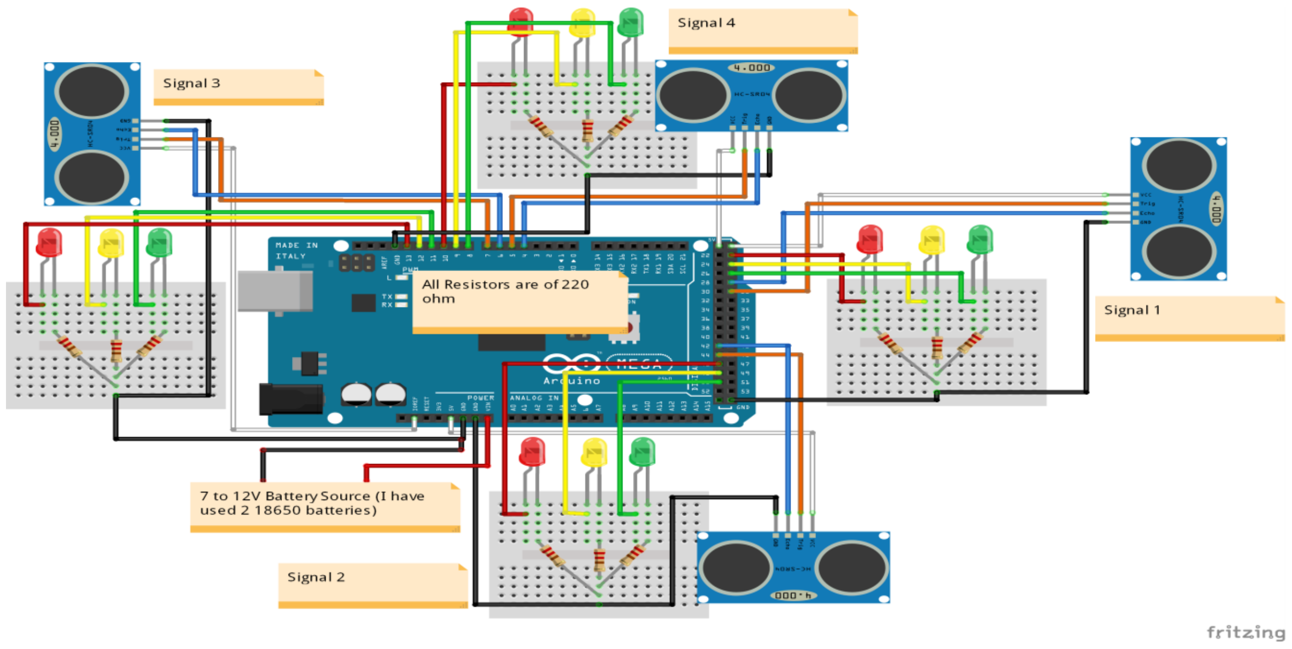
3.Based on the distance measurements, the Arduino determines if a vehicle is present within a certain range, indicating that a vehicle is waiting at the intersection.

4.If a vehicle is detected, the Arduino controls the LED lights to simulate a traffic signal:Green LED is lit to allow the vehicle to proceed.Red LED is turned off to stop traffic in the perpendicular direction.

5.The Arduino can implement timing logic for the traffic signal, adjusting the LED states based on the presence or absence of vehicles.

6.The system can communicate with a central control system or other Arduino boards at different intersections to coordinate traffic management actions based on the data collected.

CIRCUIT DIAGRAM:


Signal 4
Signal 3
Signal 1
All Resistors are of 220 ohm
7 to 12V Battery Source (I have used 2 18650 batteries)
Signal 2
fritzing

PROGRAM:

```
#include<TimerOne.h>


Int signal1[] = {23, 25, 27};

Int signal2[] = {46, 48, 50};

Int signal3[] = {13, 12, 11};

Int signal4[] = {10, 9, 8};


Int redDelay = 5000;

Int yellowDelay = 2000;


Volatile int triggerpin1 = 31;

Volatile int echopin1 = 29;

Volatile int triggerpin2 = 44;

Volatile int echopin2 = 42;
Volatile int triggerpin3 = 7;

Volatile int echopin3 = 6;

Volatile int triggerpin4 = 5;

Volatile int echopin4 = 4;


Volatile long time;                    // Variable for storing the time traveled

Volatile int S1, S2, S3, S4;           // Variables for storing the distance covered


Int t = 5;  // distance under which it will look for vehicles.


Void setup(){

  Serial.begin(115200);

  Timer1.initialize(100000);  //Begin using the timer. This function must be called first. "microseconds"
is the period of time the timer takes.

  Timer1.attachInterrupt(softInterr); //Run a function each time the timer period finishes.


  // Declaring LED pins as output

  For(int i=0; i<3; i++){

    pinMode(signal1[i], OUTPUT);

    pinMode(signal2[i], OUTPUT);

    pinMode(signal3[i], OUTPUT);

    pinMode(signal4[i], OUTPUT);

  }


  // Declaring ultrasonic sensor pins as output

  pinMode(triggerpin1, OUTPUT);

  pinMode(echopin1, INPUT);

  pinMode(triggerpin2, OUTPUT);

  pinMode(echopin2, INPUT);
```

```
 pinMode(triggerpin3, OUTPUT);
 pinMode(echopin3, INPUT);
 pinMode(triggerpin4, OUTPUT);
 pinMode(echopin4, INPUT);
}

Void loop()
{
 // If there are vehicles at signal 1
 If(S1<t)
 {
  Signal1Function();
 }

 // If there are vehicles at signal 2
 If(S2<t)
 {
  Signal2Function();
 }

 // If there are vehicles at signal 3
 If(S3<t)
 {
  Signal3Function();
 }

 // If there are vehicles at signal 4
 If(S4<t)
 {
  Signal4Function();
 }
}

// This is interrupt function and it will run each time the timer period finishes. The timer period is set
at 100 milli seconds.
Void softInterr()
{
 // Reading from first ultrasonic sensor
 digitalWrite(triggerpin1, LOW);
 delayMicroseconds(2);
 digitalWrite(triggerpin1, HIGH);
 delayMicroseconds(10);
 digitalWrite(triggerpin1, LOW);
 time = pulseIn(echopin1, HIGH);
 S1= time*0.034/2;

 // Reading from second ultrasonic sensor
 digitalWrite(triggerpin2, LOW);
 delayMicroseconds(2);
 digitalWrite(triggerpin2, HIGH);
 delayMicroseconds(10);
 digitalWrite(triggerpin2, LOW);
 time = pulseIn(echopin2, HIGH);
 S2= time*0.034/2;

 // Reading from third ultrasonic sensor
 digitalWrite(triggerpin3, LOW);
 delayMicroseconds(2);
 digitalWrite(triggerpin3, HIGH);

 delayMicroseconds(10);

 digitalWrite(triggerpin3, LOW);

 time = pulseIn(echopin3, HIGH);

 S3= time*0.034/2;


 // Reading from fourth ultrasonic sensor

 digitalWrite(triggerpin4, LOW);

 delayMicroseconds(2);

 digitalWrite(triggerpin4, HIGH);

 delayMicroseconds(10);

 digitalWrite(triggerpin4, LOW);

 time = pulseIn(echopin4, HIGH);

 S4= time*0.034/2;


 // Print distance values on serial monitor for debugging

 Serial.print("S1: ");

 Serial.print(S1);

 Serial.print("  S2: ");

 Serial.print(S2);

 Serial.print("  S3: ");

 Serial.print(S3);

 Serial.print("  S4: ");

 Serial.println(S4);

}


Void signal1Function()

{

 Serial.println("1");
 Low();
 // Make RED LED LOW and make Green HIGH for 5 seconds
 digitalWrite(signal1[0], LOW);
 digitalWrite(signal1[2], HIGH);
 delay(redDelay);

 // if there are vehicels at other signals
 If(S2<t || S3<t || S4<t)
 {
   // Make Green LED LOW and make yellow LED HIGH for 2 seconds
   digitalWrite(signal1[2], LOW);
   digitalWrite(signal1[1], HIGH);
   delay(yellowDelay);
 }
}

Void signal2Function()
{
 Serial.println("2");
 Low();
 digitalWrite(signal2[0], LOW);
 digitalWrite(signal2[2], HIGH);
 delay(redDelay);

 if(S1<t || S3<t || S4<t)
 {
   digitalWrite(signal2[2], LOW);
   digitalWrite(signal2[1], HIGH);
   delay(yellowDelay);
 }
}

Void signal3Function()
{
 Serial.println("3");
 Low();
 digitalWrite(signal3[0], LOW);
 digitalWrite(signal3[2], HIGH);
 delay(redDelay);

 if(S1<t || S2<t || S4<t)
 {
   digitalWrite(signal3[2], LOW);
   digitalWrite(signal3[1], HIGH);
   delay(yellowDelay);
 }
}

Void signal4Function()
{
 Serial.println("4");
 Low();
 digitalWrite(signal4[0], LOW);
 digitalWrite(signal4[2], HIGH);
 delay(redDelay);

 if(S1<t || S2<t || S3<t)
 {
   digitalWrite(signal4[2], LOW);
   digitalWrite(signal4[1], HIGH);
   delay(yellowDelay);
 }
}

// Function to make all LED’s LOW except RED one’s.
Void low()
{
 For(int i=1; i<3; i++)
 {
   digitalWrite(signal1[i], LOW);
   digitalWrite(signal2[i], LOW);
   digitalWrite(signal3[i], LOW);
   digitalWrite(signal4[i], LOW);
 }
 For(int i=0; i<1; i++)
 {
   digitalWrite(signal1[i], HIGH);
   digitalWrite(signal2[i], HIGH);
   digitalWrite(signal3[i], HIGH);
   digitalWrite(signal4[i], HIGH);
 }
}
```