Phase 5 : Project Documentation and submission

Project title	IOT Traffic Management System
Name	R.Subraja
Team id	5237
Team Name	Proj_204181_Team_2
College code Name	9238-Mangayarkarasi College of Engineering
Group	5
Gituhub respository link	https://github.com/Supraja1508/Ibm-Naanmudhalvan-IOT.git

IOT-TRAFFIC MANAGEMENT SYSTEM

Project objective:

Traffic management systems play a critical role in addressing the growing challenges of urban congestion, safety, and sustainability. This abstract provides an overview of key aspects and innovations in traffic management.Effective traffic management involves the integration of technologies such as smart traffic lights, real-time data analysis, and to optimize traffic flow and reduce traffic congestion.

This IoT-based Traffic Management System (IoT-TMS) leverages real-time data acquisition and analysis from various sensors and devices deployed across urban road networks. These sensors include **cameras, GPS trackers, vehicle detectors, and environmental sensors**. The collected data is transmitted to a central control system through wireless communication protocols.The central control system processes this data to monitor traffic conditions, detect congestion, and optimize traffic signal timings in real time. Additionally, IoT-TMS provides valuable insights for urban planners and policymakers by analyzing historical traffic patterns and suggesting infrastructure improvements.

2.Proposal of the project:

Development and implementation of Traffic Management System Deploy a network of sensors and devices, including Cameras, GPS trackers, vehicle detectors, and environmental sensors, across the road network.Develop user-friendly

Mobile apps to provide real-time traffic information, navigation assistance, and alerts to drivers.

2.1 Scope of work

2.1.1 Sensor Integration:

➢ We have plan to use the ultrasonic sensor for detect the presence and distance of vehicles

➢ We have plan to implement the infrared sensor to detect the vehicles

➢ We have plan to integrate camera for image processing to monitor the vehicles

➢ We have a plan to employ a raspberry pi microcontrollers to transmit the data to the cloud management

➢ We have a plan to use LoRa sensors for long-range data transmission to the cloud efficiency .

Mobile Application Development:

➢ We have a Developed a mobile app for traffic management can be a valuable tool to provide real-time

Information, alerts, and navigation assistance to drivers and pedestrians.

➢ We Provide users with real-time traffic flow information, congestion alerts, and alternative routes.

➢ We Offer turn-by-turn navigation with voice guidance and live traffic data.

2.1.3 Sensor Placement and Network Deployment:

➢ Sensors are placed in on side of the road with low traffic volume, and are used for recall control to

Change the traffic light on the side road.

➢ Sensors are weather resistant.

➢ We set up a wireless communication network for sensors to transmit data to a center hub.

2.1.4. Data Processing and Cloud Integration:

➢ We develop a machine learning algorithms for processing data from Ultrasonic, infrared sensors, and cameras and other sensors.

➢ We implement cloud-based data management for real-time analytics and Remote management.

➢ We Select cloud IoT Watson for data storage and processing the data from there the information needed By the user are transmitted.

2.1.5 User Interface and Experience:

➢ We have a plan to design an intuitive and user-friendly interface for theMobile app.

➢ It Incorporate maps, markers, search functionality, and navigation features to Guide user

➢ This app will give real-time updates on traffic congestion to drivers and pedestrians.

2.1.6.Testing and Deployment:

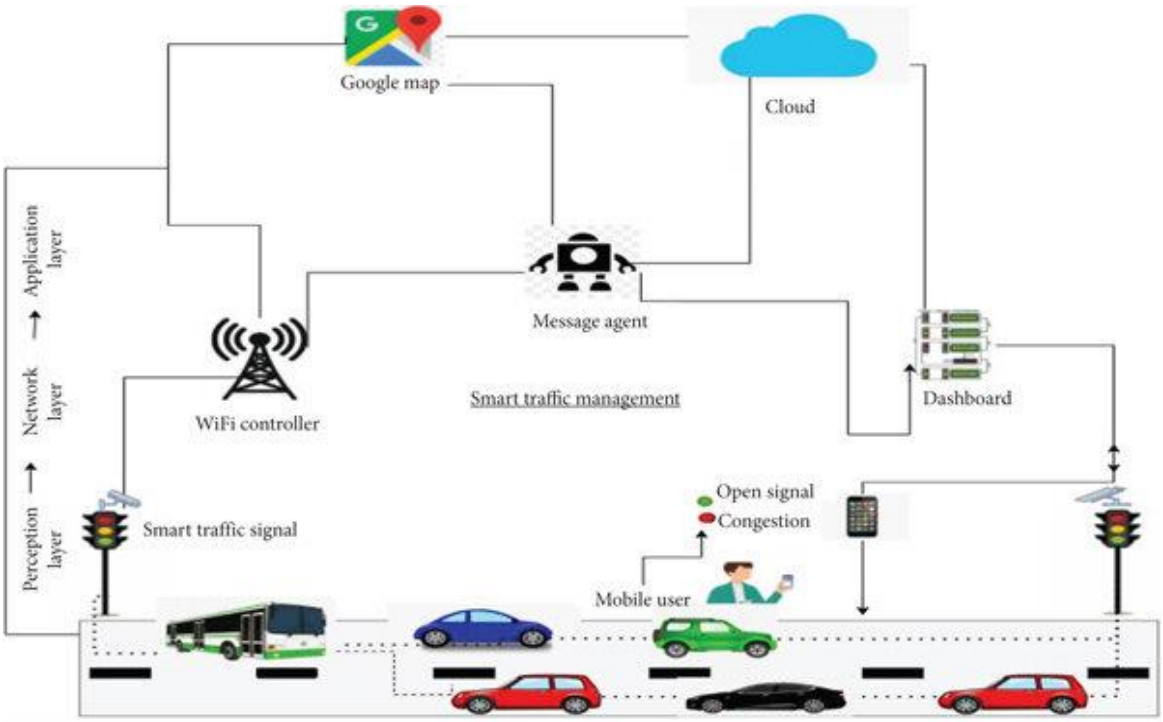
➢ We will conduct a testing of the integrated system, including sensor data Accuracy, cloud-based data

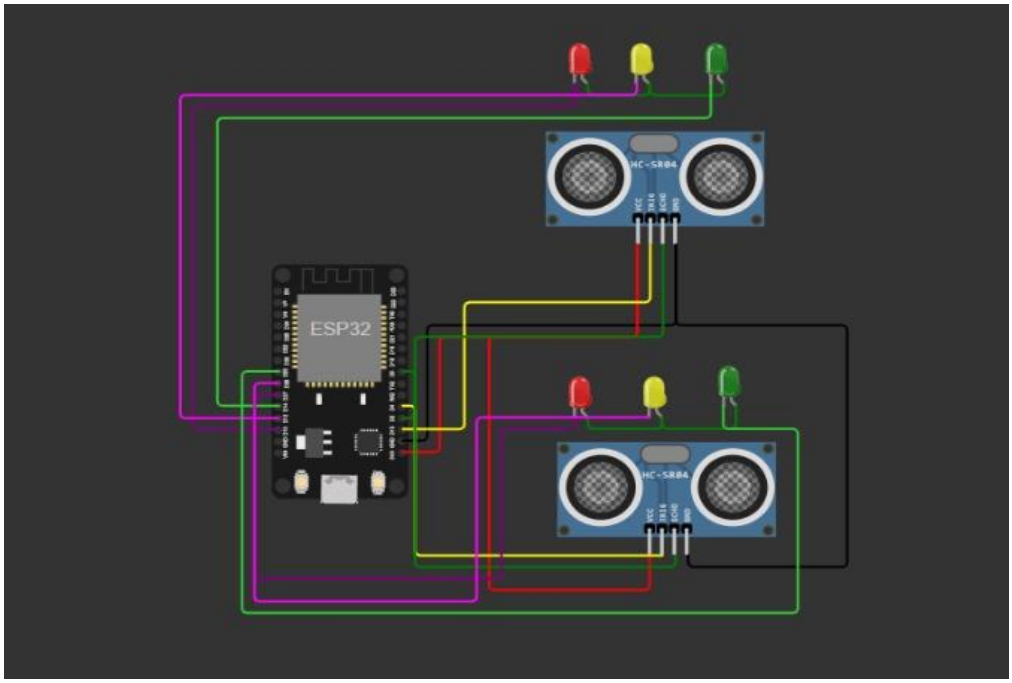
Processing, and mobile app functionality.

➢ When the testing is over and the device is run perfectly we deploy the system

➢ In road, by ensuring sensor placement and connectivity are optimal.

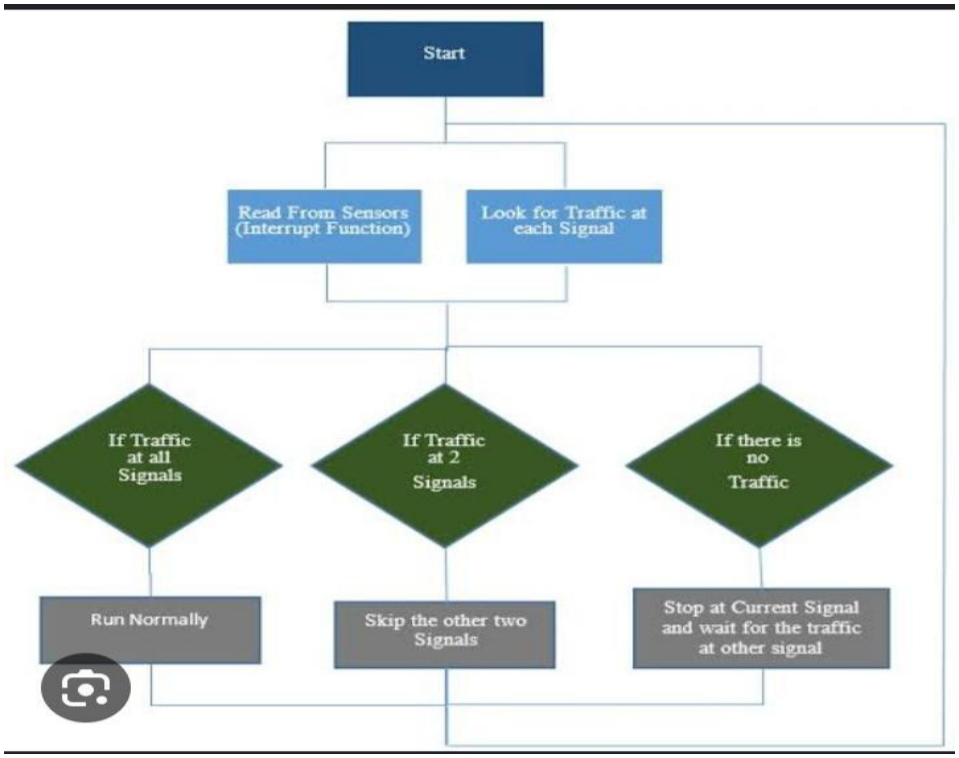
3.Processing Diagram:


Google map
Cloud
Application layer
Network layer
Perception layer
Message agent
WiFi controller
Smart traffic management
Dashboard
Open signal
Congestion
Smart traffic signal
Mobile user


ESP32
HC-SR04
HC-SR04

Here initially the data from the sensors are Collected by the micro controller then it transmit. The data to the Cloud Through the LoRa Gateway which help to transmit the data to longRange. Later the data are collected by the Cloud and Pre-Processed. It Train it self to know the is there are not by using machine learning it gets the real time update from the Sensors and Then it send the efficient information to the user through the mobile application.

4.Block Diagram:



Uses of Sensors:

1 Ultrasonic Sensor:

- **USE:**Ultrasonic sonic senor used for detect the distance of the vehicles

How they work: They emit high-frequency sound waves and measure their reflections. They detect vehicles travelling in a particular direction using a change in frequency (the Doppler effect) according to the speed of the vehicle.the sound waves Bounce off the Vehicle and return to the sensor.

● **Why we use them:**They can detect any obstacle within a distance range of a few of meters, Including both vehicles and pedestrians. Ultrasonic sensors are inexpensive and their Hardware is simple, compared with other systems

6.2 Infrared Sensor:

• **Use**:Infrared sensors can detect the presence of vehicles at intersections, toll Booths, or along roadways. This information is used to optimize traffic signal timing, Reduce congestion, and improve traffic flow.

• **How they work**:They detect the presence of vehicles by emitting and Receiving infrared Radiation. When a vehicle blocks the infrared beam, this Sensor are used.

• **Why we use them**:Infrared sensors are particularly useful in situations Where Ultrasonic sensors alone may not provide accurate readings. Together, They enhance The system's accuracy.

6.3 Camera:

• Use:Cameras serve multiple purposes, including security, license plate Recognition, And image processing.

• How they work:Cameras capture real-time images or video footage of road.

• Why we use them: Surveillance and Monitoring: Traffic cameras are strategically placed at intersections, Highways, and other critical locations to continuously monitor traffic conditions. This Real-time data helps traffic management authorities

assess traffic flow, identify Congestion, and detect incidents like accidents or

breakdowns.

• Incident Detection: Cameras are equipped with algorithms that can automatically

detect Incidents such as accidents, debris on the road, or stalled vehicles.

• Vehicle Detection and Counting: Image processing algorithms can identify and

count vehicles In realtime from camera feeds. This information helps monitor traffic

flow, assess Congestion levels, and plan traffic management strategies.

6.4 Microcontroller:

• **Use:**Microcontrollers act as the central processing units of the traffic management System.

How they work: They receive data from sensors, process it, and control Various

systemComponents.

• **Why we use them:**Microcontrollers can process data and make decisions in real-time, which isCrucial for managing traffic flow and responding to changing conditions, such as traffic Congestion or accidents.

6.5 Lora Sensor:

• **Use**: LoRa sensors are used for efficient and long-range data transmission From sensors to A central hub.

• **How They Work**: LoRa (Long Range) technology enables low-power, long-Range wireless Communication.

• **Why We Use Them:** LoRa sensors ensure seamless and reliable data.LoRa sensors can Transmit data in nearreal-time, allowing traffic management authorities to monitor traffic Conditions, detect incidents, and respond quickly to changing situations.

Sensor implementation with python coding:

```
#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 $1, $2, $3, $4;         // Variables for storing the distance covered
Int t = 5;  // distance under which it will look for vehicles.
Void setup(){
 $erial.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($1<t)
 {
   $ignal1Function();
 }
 // If there are vehicles at signal 2
 If($2<t)
 {
   $ignal2Function();
 }
 // If there are vehicles at signal 3
 If($3<t)
```

```
  {
    $ignal3Function();
  }
  // If there are vehicles at signal 4
  If($4<t)
  {
    $ignal4Function();
  }
}
// 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);
  $1= 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);
  $2= 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);
  $3= 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);
  $4= time*0.034/2;
  // Print distance values on serial monitor for debugging
  $erial.print("$1: ");
  $erial.print($1);
  $erial.print("  $2: ");
  $erial.print($2);
  $erial.print("  $3: ");
```

```
  $erial.print($3);
  $erial.print("  $4: ");
  $erial.println($4);
}
Void signal1Function()
{
  $erial.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($2<t || $3<t || $4<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()
{
  $erial.println("2");
  Low();
  digitalWrite(signal2[0], LOW);
  digitalWrite(signal2[2], HIGH);
  delay(redDelay);

  if($1<t || $3<t || $4<t)
  {
    digitalWrite(signal2[2], LOW);
    digitalWrite(signal2[1], HIGH);
    delay(yellowDelay);
  }
}
Void signal3Function()
{
  $erial.println("3");
  Low();
  digitalWrite(signal3[0], LOW);
  digitalWrite(signal3[2], HIGH);
  delay(redDelay);
  if($1<t || $2<t || $4<t)
  {
    digitalWrite(signal3[2], LOW);
    digitalWrite(signal3[1], HIGH);
    delay(yellowDelay);
  }
}
Void signal4Function()
```

```
{
  $erial.println("4");
  Low();
  digitalWrite(signal4[0], LOW);
  digitalWrite(signal4[2], HIGH);
  delay(redDelay);
  if($1<t || $2<t || $3<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);
  }
}
```

Program Description:

Library Inclusion: The code includes the "TimerOne" library to manage time intervals.

Pin Declarations: Defines various pins for LEDs (signal1, signal2, signal3, signal4), ultrasonic sensors (triggerpin1, echopin1, triggerpin2, echopin2, triggerpin3, echopin3, triggerpin4, echopin4), and Other variables like delay times.

Setup Function: This function runs once at the beginning and initializes pins, sets up the timer, and Opens a serial connection for debugging.

Loop Function: This is the main program loop that repeatedly checks if there are vehicles at each Signal and calls the appropriate function to control the traffic signal based on the detected vehicle Presence.

Interrupt Function (softInterr): This function is triggered by a timer interrupt. It reads data from four Ultrasonic sensors, calculates the distance to nearby objects, and stores this information in S1, S2, S3, and S4. It also prints these values to the serial monitor for debugging.

Firebase setup:

Realtime updates:

Firebase Realtime Database is a cloud-hosted NoSQL database provided by Firebase, a mobile and web application development platform that is now part of Google's Cloud offerings. Firebase Realtime Database is designed for real-time data Synchronization and is commonly used in applications where you need to store, Retrieve, and synchronize data across various clients and platforms.

Mobile Application Development:

Prerequisites:

1.Android Studio installed.

2. A Firebase account set up for the project.

3.An ESP32 microcontroller programmed with Micro Python.

4.A basic understanding of Python programming.

Step 1: Setting Up the Development Environment

Install Android Studio:

If not already installed, download and install Android Studio from the Official website: https://developer.android.com/studio**.**

Configure Android Studio:

Ensure you have the necessary SDKs and tools installed for Android app Development**.**

Firebase Setup:

If not done already, create a Firebase project at https://console.firebase.google.com/

and configure it for your Android app.

Step 2: Designing the App Interface

Implement the UI:

Use Android Studio's Layout Editor to create the app's user interface.

1. **Open Android Studio**: Launch Android Studio and open your Android App project.
2. **Navigate to XML Layout File:** In the project explorer, navigate to the "res" folder, then "layout," and find the XML layout file where you want To design your user interface. Double-click the XML file to open it.
3. **Open Layout Editor: O**nce you've opened the XML layout file, you'll see Two tabs at the bottom of the XML editor: "Text" and "Design." Click on The "Design" tab to open the Layout Editor.
4. **Palette**: On the left side of the Layout Editor, you'll find the "Palette" Panel. It contains various UI components such as buttons, text views, Image views, and more. You can drag and drop these components onto The layout canvas to build your interface.
5. **Component Tree:** On the right side of the Layout Editor, you'll find the "Component Tree" panel. It displays the hierarchy of UI components on Your layout. You can select and manipulate components in this panel.
6. **Attributes Panel:** Below the "Component Tree" panel, you'll find the "Attributes" panel. This panel allows you to customize the properties of Selected UI components. You can change attributes like text, color, size, And positioning.
7. **Layout Canvas:** The central area of the Layout Editor is the layout

Canvas. This is where you visually arrange and design your app's user Interface. You can drag and drop components onto the canvas, adjust Their positions, and see a real-time preview of how your layout will Appear in the app.

8. **Preview**: Above the layout canvas, there's a "Preview" panel that shows A live preview of how your layout will look on different devices and Orientations. You can switch between various screen sizes and Orientations to ensure your layout is responsive.
9. **Zoom and Pan:** You can zoom in and out of the layout canvas by using The zoom slider in the bottom right corner. You can also pan around the Canvas to work on different parts of your layout.

10.Design Toolbar: At the top of the Layout Editor, you'll find the design Toolbar. It contains options for adding constraints, aligning components, And customizing the layout.

11.Adding Constraints: Android Studio uses a constraint-based layout System (Constraint Layout) by default. To position UI components, you Can add constraints that specify how they relate to other components or The parent layout. Constraints help your layout adapt to different screen Sizes.

12.Preview Your Layout: As you design your user interface, use the "Preview" panel to see how your layout will appear on different devices And orientations. Make adjustments as needed to ensure a responsive Design.

13.Save Your Layout: Don't forget to save your layout by clicking the "Save"

Button in the top-left corner.

14.XML Code View: If you need to make fine-grained adjustments or add

Complex attributes, you can switch to the "Text" tab to edit the XML

Code directly.

Layout Program:

Activity_login.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android=http://schemas.android.com/apk/res/android
   Xmlns:app=http://schemas.android.com/apk/res-auto
   Xmlns:tools=http://schemas.android.com/tools
   Android:layout_width="match_parent"
   Android:layout_height="match_parent"
   Android:background="@drawable/traffic"
   Android:padding="16dp">

   <EditText
      Android:id="@+id/emailEditText"
      Android:layout_width="match_parent"
      Android:layout_height="wrap_content"
      Android:hint="Email"
      Android:inputType="textEmailAddress"
      Android:textColor="#F1ECEC"
      Android:textColorHint="#FFFFFF"
      App:layout_constraintBottom_toTopOf="@+id/passwordEditText"
      App:layout_constraintTop_toTopOf="parent"
      App:layout_constraintVertical_bias="0.846"
      Tools:layout_editor_absoluteX="16dp" />

   <EditText
      Android:id="@+id/passwordEditText"
      Android:layout_width="match_parent"
      Android:layout_height="wrap_content"
      Android:layout_marginBottom="436dp"
      Android:hint="Password"
      Android:inputType="textPassword"
      Android:textColorHint="#FFFFFF"
      App:layout_constraintBottom_toBottomOf="parent"
```

```
    Tools:layout_editor_absoluteX="16dp" />

  <Button
    Android:id="@+id/registerButton"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:backgroundTint="#090305"
    Android:text="Register"
    App:layout_constraintBottom_toBottomOf="parent"
    App:layout_constraintEnd_toEndOf="parent"
    App:layout_constraintHorizontal_bias="0.803"
    App:layout_constraintStart_toStartOf="parent"
    App:layout_constraintTop_toBottomOf="@+id/passwordEditText"
    App:layout_constraintVertical_bias="0.094" />

  <TextView
    Android:id="@+id/textView2"
    Android:layout_width="286dp"
    Android:layout_height="69dp"
    Android:text="IOT Air Quality Sign Up"
    Android:textAlignment="center"
    Android:textAllCaps="true"
    Android:textSize="24sp"
    Android:textStyle="bold"
    Tools:layout_editor_absoluteX="48dp"
    Tools:layout_editor_absoluteY="39dp" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android=http://schemas.android.com/apk/res/android
   Xmlns:app=http://schemas.android.com/apk/res-auto
   Xmlns:tools=http://schemas.android.com/tools
   Android:layout_width="match_parent"
   Android:layout_height="match_parent"
   Android:background="@drawable/traffic"
   Android:padding="16dp"
   Tools:ignore="ExtraText">


   <EditText
      Android:id="@+id/emailEditText"
      Android:layout_width="match_parent"
      Android:layout_height="wrap_content"
      Android:hint="Email"
      Android:inputType="textEmailAddress"
      Android:textColor="#FAFAFA"
      Android:textColorHint="#FFFFFF"
      App:layout_constraintBottom_toTopOf="@+id/passwordEditText"
      App:layout_constraintTop_toTopOf="parent"
```

```
   App:layout_constraintVertical_bias="0.812"
   Tools:layout_editor_absoluteX="16dp" />

<EditText
   Android:id="@+id/passwordEditText"
   Android:layout_width="match_parent"
   Android:layout_height="wrap_content"
   Android:layout_marginBottom="476dp"
   Android:hint="Password"
   Android:inputType="textPassword"
   Android:textColor="#F1EEEE"
   Android:textColorHighlight="#F8F5F5"
   Android:textColorHint="#F3F0F0"
   App:layout_constraintBottom_toBottomOf="parent"
   App:layout_constraintEnd_toEndOf="parent"
   App:layout_constraintHorizontal_bias="0.125"
   App:layout_constraintStart_toStartOf="parent" />

<Button
   Android:id="@+id/loginButton"
   Android:layout_width="95dp"
   Android:layout_height="49dp"
   Android:layout_marginEnd="64dp"
   Android:backgroundTint="#1938E1"
   Android:text="Login"
   Android:textColorHighlight="#CD7C7C"
   App:layout_constraintBottom_toBottomOf="parent"
   App:layout_constraintEnd_toEndOf="parent"
   App:layout_constraintTop_toTopOf="parent"
   App:layout_constraintVertical_bias="0.419"
   App:rippleColor="#E14545"
   App:strokeColor="#E85656" />

<Button
   Android:id="@+id/signup"
   Android:layout_width="101dp"
   Android:layout_height="46dp"
   Android:layout_marginEnd="24dp"
   Android:backgroundTint="#2844DF"
   Android:text="signup"
   Android:textColorHighlight="#DC4040"
   Android:textColorLink="#D84577"
   App:layout_constraintBottom_toBottomOf="parent"
   App:layout_constraintEnd_toStartOf="@+id/loginButton"
   App:layout_constraintTop_toTopOf="parent"
   App:layout_constraintVertical_bias="0.42" />

<TextView
   Android:id="@+id/textView"
   Android:layout_width="251dp"
   Android:layout_height="36dp"
```

Android:layout_marginBottom=”24dp”
Android:fontFamily=”sans-serif-black”
Android:text=”IOT traffic monitoring”
Android:textAlignment=”center”
Android:textColor=”#FAF6F6”
Android:textColorHighlight=”#FAF9F9”
Android:textColorHint=”#FFFFFF”
Android:textColorLink=”#FBF9F9”
Android:textSize=”20sp”
Android:textStyle=”bold”
App:layout_constraintBottom_toTopOf=”@+id/emailEditText”
App:layout_constraintEnd_toEndOf=”parent”
App:layout_constraintHorizontal_bias=”0.506”
App:layout_constraintStart_toStartOf=”parent”
App:layout_constraintTop_toTopOf=”parent”
App:layout_constraintVertical_bias=”0.25” />

</androidx.constraintlayout.widget.ConstraintLayout>

Program description:

Android UI Layout Description

The provided code is an XML layout file for an Android app’s user interface (UI). It’s written using the ConstraintLayout, which is a type of layout in Android that allows you to create complex and responsive UIs. Here’s a brief program description:

The root element is a ConstraintLayout, which is used to create a flexible and responsive UI. It defines various attributes such as background, padding, and layout width/height.

Within the ConstraintLayout, there are several child views:

Two EditText elements for email and password input fields.

Two Button elements for login and signup actions.

One TextView displaying the text “IOT traffic monitoring.”

Each view (EditText, Button, and TextView) has specific attributes defined, such as IDs, layout dimensions, hint text, text color, and constraints that define their position relative to other views.

The XML file also includes various styling attributes like text color, text size, font, and background color for the UI elements.

It seems to be related to an “IOT traffic monitoring” application, possibly for user authentication with email and password.

Please note that the provided XML layout is a part of an Android app's UI, and its functionality would be implemented in Java or Kotlin code. This layout defines the visual structure of the app's login or signup screen.

Activity _firebase:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android=http://schemas.android.com/apk/res/android
   Xmlns:app=http://schemas.android.com/apk/res-auto
   Xmlns:tools=http://schemas.android.com/tools
   Android:layout_width="match_parent"
   Android:layout_height="match_parent"
   Android:background="#130035"
   Android:padding="16dp">

   <TextView
      Android:id="@+id/textViewDistance"
      Android:layout_width="336dp"
      Android:layout_height="31dp"
      Android:text="Distance: 0 m"
      Android:textAlignment="center"
      Android:textAllCaps="true"
      Android:textColor="#FFFFFF"
      Android:textSize="13sp"
      App:layout_constraintBottom_toBottomOf="parent"
      App:layout_constraintEnd_toEndOf="parent"
      App:layout_constraintHorizontal_bias="0.454"
      App:layout_constraintStart_toStartOf="parent"
      App:layout_constraintTop_toTopOf="parent"
      App:layout_constraintVertical_bias="0.105" />

   <TextView
      Android:id="@+id/textViewDistance2"
      Android:layout_width="314dp"
      Android:layout_height="28dp"
      Android:text="Distance: 0 m"
      Android:textAlignment="center"
      Android:textAllCaps="true"
      Android:textColor="#FFFFFF"
      Android:textSize="13sp"
      App:layout_constraintBottom_toBottomOf="parent"
      App:layout_constraintEnd_toEndOf="parent"
      App:layout_constraintStart_toStartOf="parent"
      App:layout_constraintTop_toTopOf="parent"
      App:layout_constraintVertical_bias="0.608" />

   <ImageView
      Android:id="@+id/trafficLightImage"
      Android:layout_width="192dp"
```

```
    Android:layout_height=”167dp”

    App:layout_constraintBottom_toBottomOf=”parent”
    App:layout_constraintEnd_toEndOf=”parent”
    App:layout_constraintHorizontal_bias=”0.456”
    App:layout_constraintStart_toStartOf=”parent”
    App:layout_constraintTop_toTopOf=”parent”
    App:layout_constraintVertical_bias=”0.223” />

<ImageView
    Android:id=”@+id/trafficLightImage2”
    Android:layout_width=”192dp”
    Android:layout_height=”167dp”

    App:layout_constraintBottom_toBottomOf=”parent”
    App:layout_constraintEnd_toEndOf=”parent”
    App:layout_constraintHorizontal_bias=”0.461”
    App:layout_constraintStart_toStartOf=”parent”
    App:layout_constraintTop_toTopOf=”parent”
    App:layout_constraintVertical_bias=”0.852” />

<TextView
    Android:id=”@+id/textViewStatus”
    Android:layout_width=”wrap_content”
    Android:layout_height=”wrap_content”
    Android:shadowColor=”#FDFDFD”
    Android:text=”Status: Red”
    Android:textAlignment=”center”
    Android:textAllCaps=”true”
    Android:textColor=”#FFFFFF”
    Android:textSize=”15sp”
    App:layout_constraintBottom_toBottomOf=”parent”
    App:layout_constraintEnd_toEndOf=”parent”
    App:layout_constraintHorizontal_bias=”0.47”
    App:layout_constraintStart_toStartOf=”parent”
    App:layout_constraintTop_toTopOf=”parent”
    App:layout_constraintVertical_bias=”0.437” />

<TextView
    Android:id=”@+id/textViewStatus2”
    Android:layout_width=”wrap_content”
    Android:layout_height=”wrap_content”
    Android:shadowColor=”#FFFFFF”
    Android:text=”Status: Red”
    Android:textAlignment=”center”
    Android:textAllCaps=”true”
    Android:textColor=”#FFFFFF”
    Android:textSize=”15sp”
    App:layout_constraintBottom_toBottomOf=”parent”
    App:layout_constraintEnd_toEndOf=”parent”
    App:layout_constraintHorizontal_bias=”0.476”
```

```
    App:layout_constraintStart_toStartOf=”parent”
    App:layout_constraintTop_toTopOf=”parent”
    App:layout_constraintVertical_bias=”0.942” />

  <TextView
    Android:id=”@+id/textView3”
    Android:layout_width=”wrap_content”
    Android:layout_height=”wrap_content”
    Android:text=”Signal 1 “
    Android:textAlignment=”center”
    Android:textAllCaps=”true”
    Android:textColor=”#FFFFFF”
    Android:textSize=”20sp”
    Android:textStyle=”bold”
    App:layout_constraintBottom_toBottomOf=”parent”
    App:layout_constraintEnd_toEndOf=”parent”
    App:layout_constraintHorizontal_bias=”0.503”
    App:layout_constraintStart_toStartOf=”parent”
    App:layout_constraintTop_toTopOf=”parent”
    App:layout_constraintVertical_bias=”0.046” />

  <TextView
    Android:id=”@+id/textView4”
    Android:layout_width=”wrap_content”
    Android:layout_height=”wrap_content”
    Android:text=”Signal 2”
    Android:textAlignment=”center”
    Android:textAllCaps=”true”
    Android:textColor=”#FFFFFF”
    Android:textSize=”20sp”
    Android:textStyle=”bold”
    App:layout_constraintBottom_toBottomOf=”parent”
    App:layout_constraintEnd_toEndOf=”parent”
    App:layout_constraintHorizontal_bias=”0.495”
    App:layout_constraintStart_toStartOf=”parent”
    App:layout_constraintTop_toTopOf=”parent”
    App:layout_constraintVertical_bias=”0.542” />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Program description:

The provided XML code represents the layout of an Android app’s user interface using a ConstraintLayout. Here’s a brief program description:

The root element is a ConstraintLayout, which is a flexible layout manager for positioning and aligning UI elements.

It contains several child elements, including TextViews and ImageViews, which are used to display text and images.

There are two TextViews with IDs textViewDistance and textViewDistance2, which display the text "Distance: 0 m." They are positioned at different vertical biases within the layout.

Two ImageViews with IDs trafficLightImage and trafficLightImage2 are used to display images. They are also positioned differently within the layout.

Two TextViews with IDs textViewStatus and textViewStatus2 display the text "Status: Red." They are positioned at different vertical biases and use shadow effects for text.

Two additional TextViews, textView3 and textView4, display the text "Signal 1" and "Signal 2" respectively, with different vertical biases.

The layout elements have various attributes for specifying their dimensions, text content, text styles, colors, and constraints for positioning.

This XML layout is designed to create a user interface with text and image elements, likely for displaying information related to traffic lights or signals. It uses a ConstraintLayout to manage the positioning of these elements on the screen.

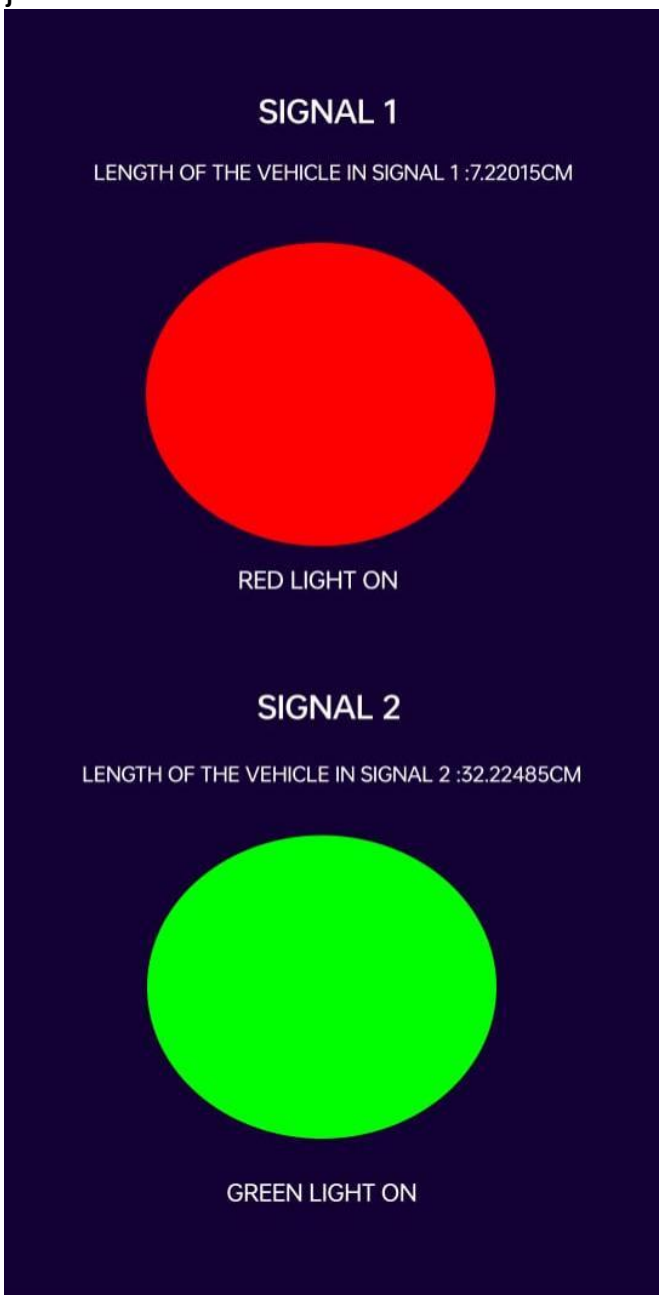
Java program:

```
Package com.example.smarttrafficmanagement;

Import android.annotation.SuppressLint;
Import android.os.Bundle;
Import android.util.Log;
Import android.widget.ImageView;
Import android.widget.TextView;
Import android.widget.Toast;

Import androidx.annotation.NonNull;
Import androidx.appcompat.app.AppCompatActivity;

Import com.google.firebase.database.DataSnapshot;
Import com.google.firebase.database.DatabaseError;
Import com.google.firebase.database.DatabaseReference;
Import com.google.firebase.database.FirebaseDatabase;
Import com.google.firebase.database.ValueEventListener;

Public class Firebase extends AppCompatActivity {

   Private DatabaseReference sensorDataRef,sensorDataRef1;

   Private TextView textViewDistance;
   Private TextView textViewDistance2;

   Private TextView status,status1;

   Private TextView  trafficTextView;
Private ImageView img1,img2;
```

```
@SuppressLint("CutPasteId")
@Override
Protected void onCreate(Bundle savedInstanceState) {
  Super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_firebase);

  // Initialize Firebase
  FirebaseDatabase firebaseDatabase = FirebaseDatabase.getInstance();
  sensorDataRef = firebaseDatabase.getReference("TrafficLightStatus1");
  sensorDataRef1 = firebaseDatabase.getReference("TrafficLightStatus2");
  // Get references to GaugeView widgets in your layout
  textViewDistance=findViewById(R.id.textViewDistance);
  textViewDistance2=findViewById(R.id.textViewDistance2);
  status=findViewById(R.id.textViewStatus);
  status1=findViewById(R.id.textViewStatus2);
  img1=findViewById(R.id.trafficLightImage);
  img2=findViewById(R.id.trafficLightImage2);



  sensorDataRef.addValueEventListener(new ValueEventListener() {
    @SuppressLint("SetTextI18n")
    @Override
    Public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
      If (dataSnapshot.exists()) {
        // Retrieve values from the dataSnapshot
        Double Distance = dataSnapshot.child("Distance1").getValue(Double.class);
        Integer redstatus = dataSnapshot.child("Red").getValue(Integer.class);
        Integer yelstatus = dataSnapshot.child("Yellow").getValue(Integer.class);
        Integer grnstatus = dataSnapshot.child("Green").getValue(Integer.class);


        textViewDistance.setText("Length of the Vehicle in Signal 1
:"+String.valueOf(Distance)+"CM");

        if(redstatus==1)
        {
          Status.setText("RED LIGHT ON");
          Img1.setBackgroundResource(R.drawable.red_circle);

        }
        If(yelstatus==1)
        {
          Status.setText("YELLOW LIGHT ON");
          Img1.setBackgroundResource(R.drawable.yellow_circle);

        }
        If(grnstatus==1)
        {
          Status.setText("Green LIGHT ON");
          Img1.setBackgroundResource(R.drawable.green_circle);
```

```
            }



          }

        }

        @Override
        Public void onCancelled(@NonNull DatabaseError databaseError) {
          // Handle database error
        }
      });
      sensorDataRef1.addValueEventListener(new ValueEventListener() {
        @SuppressLint("SetTextI18n")
        @Override
        Public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
          If (dataSnapshot.exists()) {
            // Retrieve values from the dataSnapshot
            Double Distance = dataSnapshot.child("Distance2").getValue(Double.class);
            Integer redstatus1 = dataSnapshot.child("Red").getValue(Integer.class);
            Integer yelstatus1 = dataSnapshot.child("Yellow").getValue(Integer.class);
            Integer grnstatus1 = dataSnapshot.child("Green").getValue(Integer.class);


            textViewDistance2.setText("Length of the Vehicle in Signal 2
:"+String.valueOf(Distance)+"CM");

            if(redstatus1==1)
            {
              Status1.setText("RED LIGHT ON");
              Img2.setBackgroundResource(R.drawable.red_circle);

            }
            If(yelstatus1==1)
            {
              Status1.setText("YELLOW LIGHT ON");
              Img2.setBackgroundResource(R.drawable.yellow_circle);

            }
            If(grnstatus1==1)
            {
              Status1.setText("Green LIGHT ON");
              Img2.setBackgroundResource(R.drawable.green_circle);
            }



          }

        }
```

```
        @Override
        Public void onCancelled(@NonNull DatabaseError databaseError) {
            // Handle database error
        }
    });

}
```



Conclusion:

In conclusion, the traffic management project represents a crucial step toward enhancing transportation efficiency, safety, and sustainability in our community. By implementing a

combination of smart traffic signals, data analytics, and public awareness campaigns, we aim to reduce congestion, minimize accidents, and promote eco-friendly modes of transport. While challenges may arise, continuous monitoring and adaptation of the system will be essential for long-term success. Ultimately, this project is poised to make a positive impact on the quality of life for residents and visitors alike.