TRAFFIC MANAGEMENT SYSTEM

**INTRODUCTION:** The traditional methods of managing urban mobility in this day and age are showing their limitations. Additionally, the demand for an effective traffic control system is rising. Thus, we are controlling the traffic system with IOT technology.

# PROJECT SCOPE FEATURES:

**Real-time Traffic Monitoring**: Continuous monitoring of traffic conditions, including vehicle counts, speeds, and congestion levels, through various sensors and cameras.

**Traffic Data Analytics**: Advanced data analytics to process and analyze real- time data, identifying traffic patterns, congestion hotspots, and historical trends.

**Parking Availability Apps**: Apps that inform drivers about available parking spaces in real-time, reducing the time spent searching for parking and minimizing traffic congestion caused by circling vehicles.

**Emergency Assistance through IoT:** A traffic monitoring system using IoT technology enables emergency responders to speed up the care mechanism in case of accidents late at night or in isolated locations. The sensors on the road detect any accident, and the problem is immediately reported to the traffic management system. This request is passed on to relevant authorities to take corrective action. Emergency response personnel would include medical technicians, police officers, and fire departments for enhanced responsiveness and timely intervention.

**Commute Assistance:** With every vehicle acting as an IoT sensor, a dedicated app can make suggestions, determine optimal routes & provide advance notice of accidents or traffic jams. Further, it can even suggest the best time to leave. It is all because of a robust algorithm that helps reduce driving time with intelligent traffic lights**.**

# Technologies used:

**User applications** that allow citizens to receive instant notifications in case of traffic jams and congested routes. Desktop user apps for control rooms send commands to actuators for altering traffic signals. It helps to relieve congestion and optimize routes.

**Cross-solution integrations** with traffic lights or streetlight management systems. Control apps apply ML models or predefined rules to prompt appropriate output action if the air quality is poor.

**Data analytics** for analyzing the data from streetlight sensors on a centralized dashboard to adjust the intensity of lights

**ML algorithms** to analyze traffic patterns & trends from historical data – stored in the data warehouse. The identified trends are then used to build predictive models for control apps. These apps modify the average vehicle speed to avoid congestion

# Real-time Data Integration:

Real-time restroom data is integrated via Firebase Realtime Database API, ensuring instant updates. Data synchronization between web and mobile platforms is achieved through Firebase's real-time syncing mechanism, guaranteeing consistency. Robust error handling protocols are implemented, with data consistency measures in place, guaranteeing users accurate and reliable restroom availability and cleanliness information at all times.

# Online Stimulation:

This project integrates Wowki, an IoT platform, for online stimulation purposes. Wowki facilitates seamless connectivity and interaction between IoT devices, enabling efficient communication, data exchange, and remote control.

Leveraging Wowki, the project enhances IoT experiences, ensuring real-time stimulation and monitoring capabilities for various applications in the Internet of Things ecosystem.

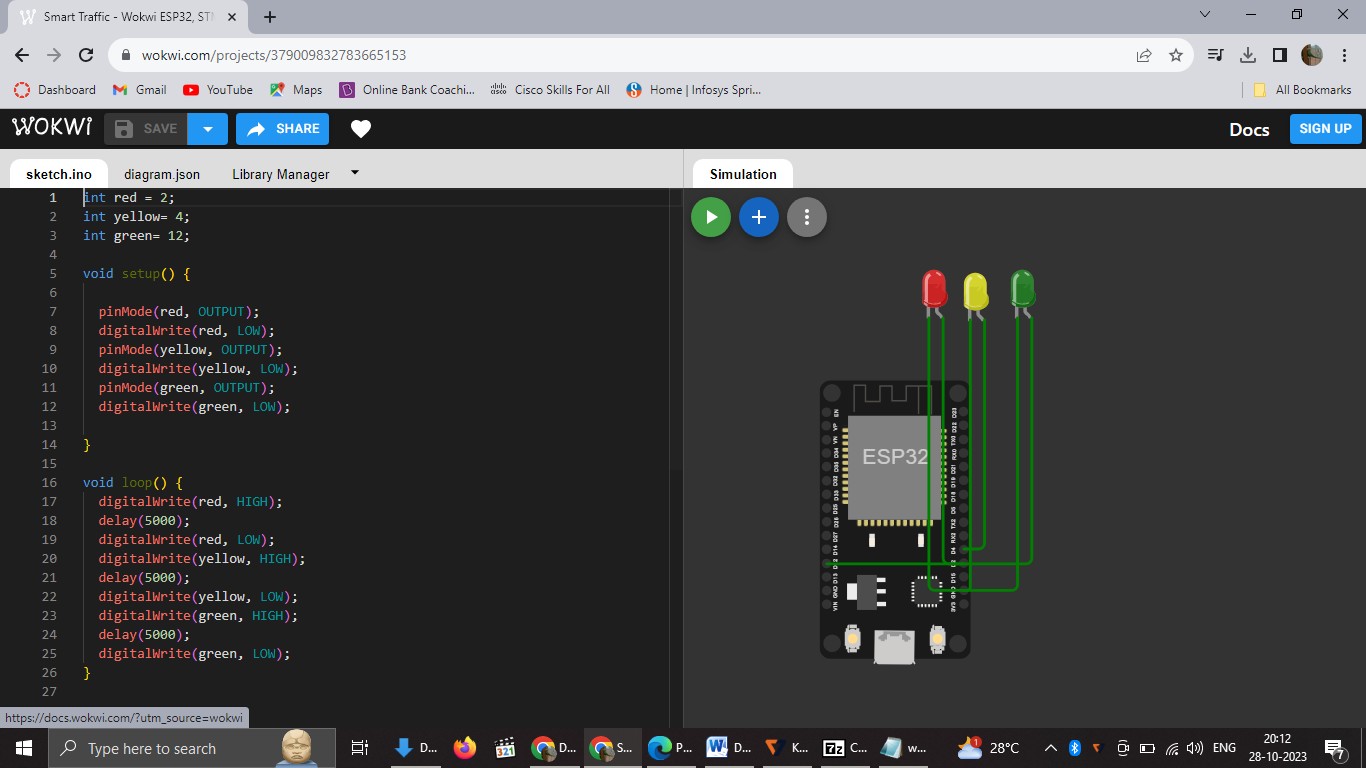


Fig Running the code for traffic management system

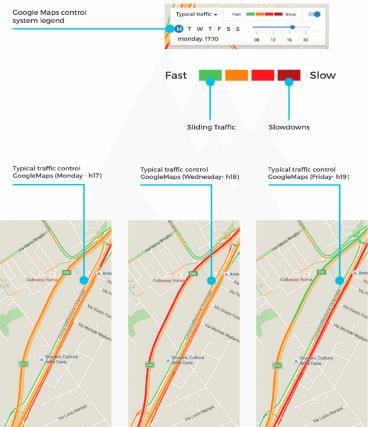


Fig Sample Page of app interface

1. **BUILDING THE IOT TRAFFIC MONITORING SYSTEM:**

**STEP 1: IoT DEVICE DEPLOYMENT AND DATA COLLECTION:**

import random

import time

import requests

def collect\_traffic\_data():

while True:

vehicle\_count = random.randint(0, 100)

vehicle\_speed = random.uniform(10, 120)

vehicle\_density = random.uniform(0, 1)

send\_data\_to\_platform(vehicle\_count, vehicle\_speed, vehicle\_density)

time.sleep(5)

def send\_data\_to\_platform(count, speed, density):

endpoint = "http://your\_traffic\_platform\_endpoint"

payload = {"vehicle\_count": count, "vehicle\_speed": speed, "vehicle\_density": density}

try:

response = requests.post(endpoint, json=payload)

if response.status\_code == 200:

print("Data sent successfully.")

else:

print("Failed to send data. Status code:", response.status\_code)

except requests.exceptions.RequestException as e:

print("Error:", e)

if \_\_name\_\_ == '\_\_main\_\_':

collect\_traffic\_data()

**output:**

**Data sent successfully.**

**Data sent successfully.**

**Data sent successfully.**

**Data sent successfully.**

**...**

**EXPLANATION:**

* This Python script simulates actual-time traffic facts collection by producing random automobile matter, pace, and density values.
* Replace the simulated facts with actual data received from IoT gadgets (traffic glide sensors, cameras).
* The script sends the amassed statistics to the visitors statistics platform via an HTTP POST request.

**STEP2: INTEGRATING IOT DATA WITH TRAFFIC INTELLIGENCE PLATFORM:**

def interpret\_traffic\_data(traffic\_data, iot\_data):

combined\_data = {

"vehicle\_count": iot\_data["vehicle\_count"],

"vehicle\_speed": iot\_data["vehicle\_speed"],

"vehicle\_density": iot\_data["vehicle\_density"],

"congestion\_level": calculate\_congestion\_level(iot\_data)

}

return combined\_data

def calculate\_congestion\_level(iot\_data):

if iot\_data["vehicle\_density"] > 0.7:

return "High congestion"

elif iot\_data["vehicle\_density"] > 0.4:

return "Moderate congestion"

else:

return "Low congestion"

output

{

"vehicle\_count": 75,

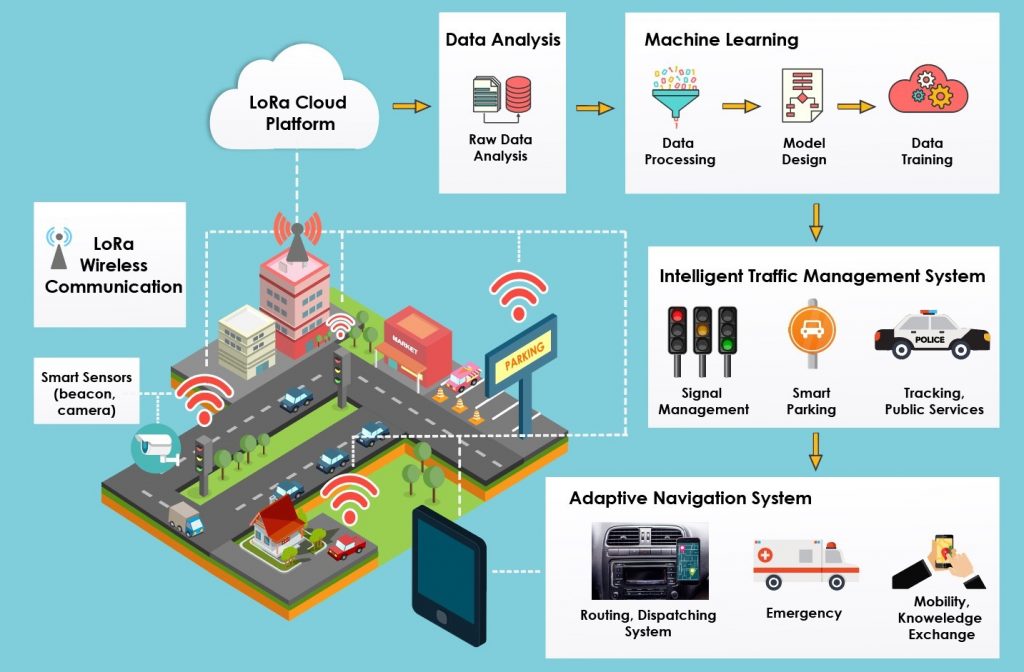
"vehicle\_speed": 65.2,

"vehicle\_density": 0.6,

"congestion\_level": ""

}





**EXPLANATION:**

* Modify the existing interpret\_traffic\_data characteristic to combine IoT information (car rely, pace, density) into the site visitors intelligence platform.
* Calculate the congestion level based on the IoT information. Now, the IoT traffic tracking system is built and included with the visitor’s intelligence platform, allowing real-time site visitors data collection and processing for better traffic control and predictions.

**STEP3: PYTHON SCRIPT FOR REAL-TIME DATA TRANSMISSION:**

import random

import time

import requests

def collect\_traffic\_data():

while True:

vehicle\_count = random.randint(0, 100)

vehicle\_speed = random.uniform(10, 120)

vehicle\_density = random.uniform(0, 1)

send\_data\_to\_platform(vehicle\_count, vehicle\_speed, vehicle\_density)

time.sleep(5) # Simulate data collection every 5 seconds

def send\_data\_to\_platform(count, speed, density):

endpoint = "http://your\_traffic\_platform\_endpoint"

payload = {

"vehicle\_count": count,

"vehicle\_speed": speed,

"vehicle\_density": density

}

Try:

response = requests.post(endpoint, json=payload)

if response.status\_code == 200:

print("Data sent successfully.")

else:

print ("Failed to send data. Status code:", response.status\_code)

except requests.exceptions.RequestException as e:

print("Error:", e)

if \_\_name\_\_ == '\_\_main\_\_':

collect\_traffic\_data()

**EXPLANATION:**

* The Python script simulates actual-time visitors facts collection by producing random car matter, velocity, and density values.
* Replace the simulated records with actual statistics acquired from IoT gadgets (visitors drift sensors, cameras).
* The script sends the amassed facts to the traffic information platform through an HTTP POST request.
* By deploying IoT gadgets and jogging the Python script on those gadgets, actual-time visitor’s statistics may be accumulated and transmitted to the visitors records platform, enabling better traffic tracking and management.

