TRAFFIC MANAGEMENT

Phase 3: Development part 1

TRAFFIC MANAGEMENT SYSTEM – IOT

Phase 3: Development part 1

Introduction:

The sustainability and smartness of the smart city concept rely on the technologies adopted to improve the people's quality of life. The smart city governance is one significant aspect of smart city initiatives, which will facilitate the planning techniques for better decision making. One of the key elements of the smart city governance framework is the public value generated out of the smart services provided.

The government has to work on different aspects of smart city solutions such as smart health care, smart building management, smart traffic management, smart parking solutions, smart transportation, etc. to generate public value for the service they provided. The emergence of the internet of things (IoT) has evolved the concept of smart cities. In a smart city environment, the physical infrastructures of the city are equipped with smart devices, which continuously produce multidimensional data in different spaces and these data are processed to achieve intelligence for the infrastructure. Ultimately, intelligence is applied to improve the socio-economic activities of the society.

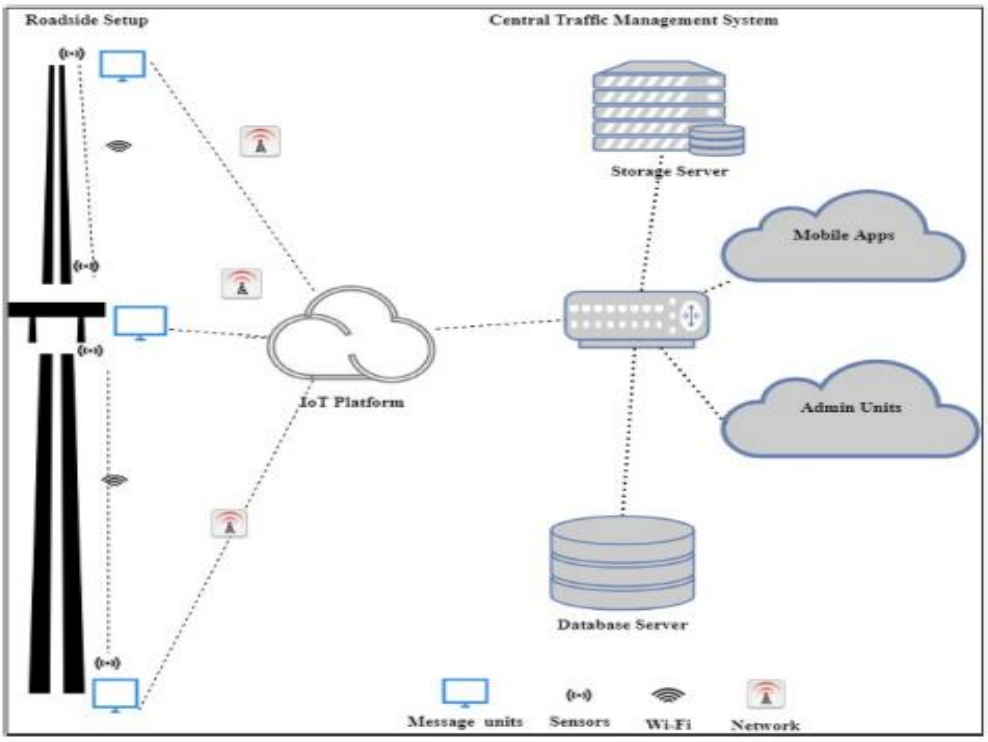
Smart traffic infrastructure is an essential component of smart city initiatives because traffic congestion is a severe issue that grows along with city development. Smart traffic management includes intelligent transport systems with integrated components like adaptive traffic signal controls, freeway management, emergency management services, and roadside units.

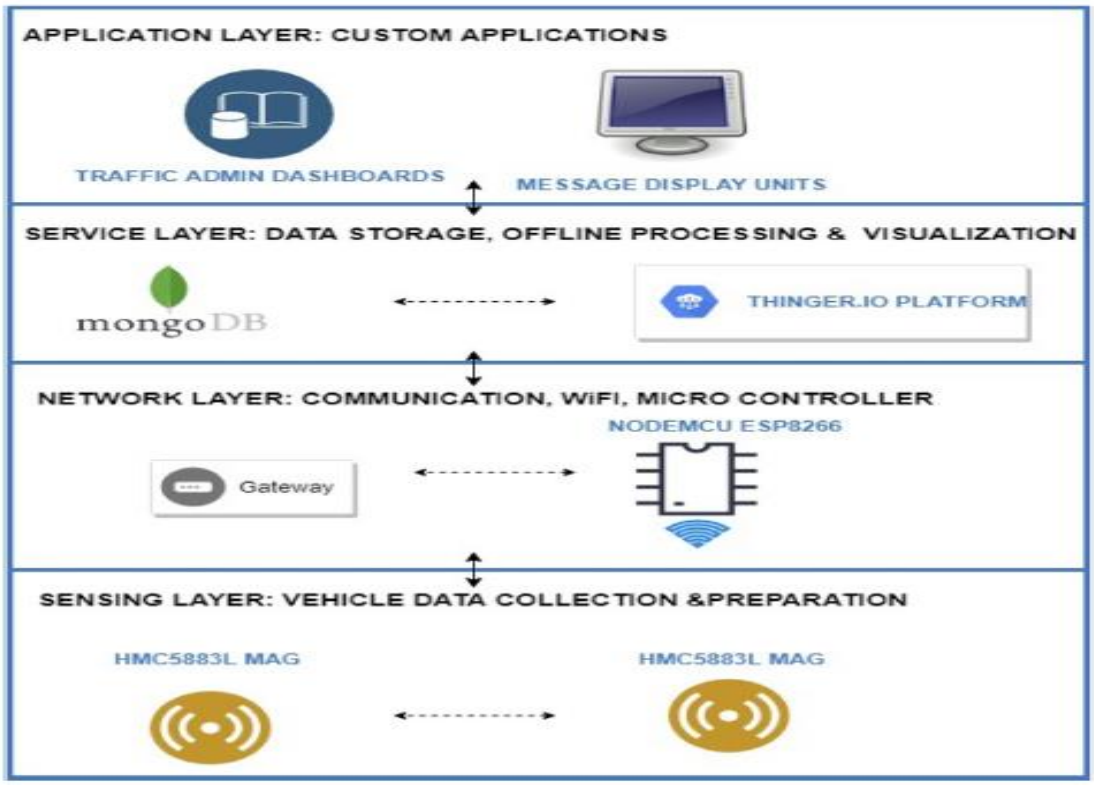
System Design and Development:

This section discusses the proposed system model, different software and hardware components required, and algorithms to implement the proposed system. The proposed system communication model is presented, which has components installed at the roadside and a cloud-based central server. The roadside setup includes sensors and message boards. The sensors and boards will be installed between two road segment intersections. The central server includes data storage, cloud services, and interfaces. The components can communicate with each other using WIFI.

System architecture:

An IoT based system architecture mostly contains a sensing layer, network layer, service layer, and an application layer. The sensing layer acquires data from the things, the network layer transfers the collected data from devices to the service layer, the service layer controls the devices and analyzes the collected data, and finally, the application layer which indicates the user interface. The layered architecture is presented. The four main system development activities are: (i) populate geographical map details for a given location, (ii) detect vehicle and estimate vehicle length, (iii) determine growing queue, and (iv) display traffic updates.


Roadside Setup
Central Traffic Management System
Storage Server
Mobile Apps
IoT Platform
Admin Units
Database Server
Message units
Sensors
Wi-Fi
Network


APPLICATION LAYER: CUSTOM APPLICATIONS
TRAFFIC ADMIN DASHBOARDS
MESSAGE DISPLAY UNITS
SERVICE LAYER: DATA STORAGE, OFFLINE PROCESSING & VISUALIZATION
mongoDB
THINGER.IO PLATFORM
NETWORK LAYER: COMMUNICATION, WiFI, MICRO CONTROLLER
NODEMCU ESP8266
Gateway
SENSING LAYER: VEHICLE DATA COLLECTION &PREPARATION
HMC5883L MAG
HMC5883L MAG

Loading and preprocessing datasets in traffic management system using IoT:

1. Load Data:

- Use appropriate libraries (like pandas in Python) to load the dataset from a source (e.g., CSV, database, API).

2. Data Cleaning:

- Handle Missing Data: Determine how to deal with missing values. Options include removing rows/columns with missing values or imputing them with values (mean, median, mode, or using more advanced methods).

- Remove Duplicates: Identify and remove any duplicate entries to prevent skewed analysis.

3. Date-Time Processing:

- Convert string timestamp entries to a recognized date-time format. This enables easier temporal analysis and aggregations.

- Extract useful time components like hour, day, week, etc., which can be useful for further analyses.

4. Data Filtering:

- Identify and remove any unrealistic or erroneous data. For traffic data, examples could be speeds that are beyond feasible limits or negative vehicle counts.

5. Data Transformation:

- Categorization: Convert continuous data to categorical data if needed. For instance, categorizing speeds into 'slow', 'medium', and 'fast'.

- Normalization or Standardization: Transform features to be on a common scale, especially if you're planning to use machine learning algorithms that are sensitive to feature scales.

6. Data Aggregation:

- Group data based on specific criteria to gain insights. For instance, you can group by hour to understand traffic patterns during different times of the day.

7. Feature Engineering:

- Create new informative features from the existing data, which can be helpful for analysis or modeling. Examples include traffic growth rate, average speed changes, etc.

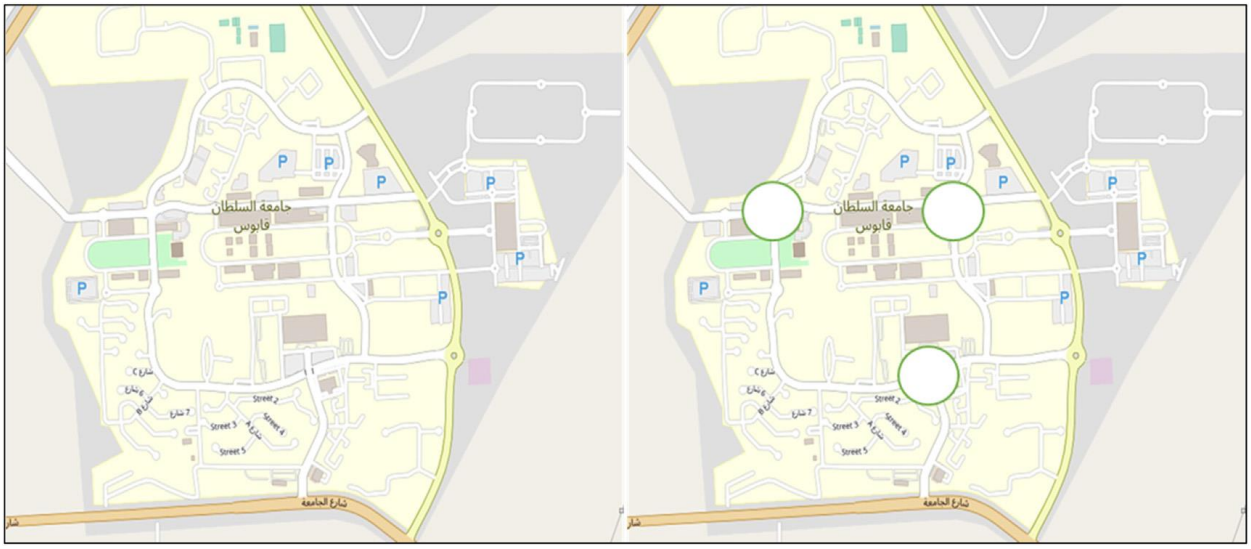
8. Store Processed Data:

- Save the preprocessed data to a new file, database, or another storage solution for easier access in subsequent analyses or operations.

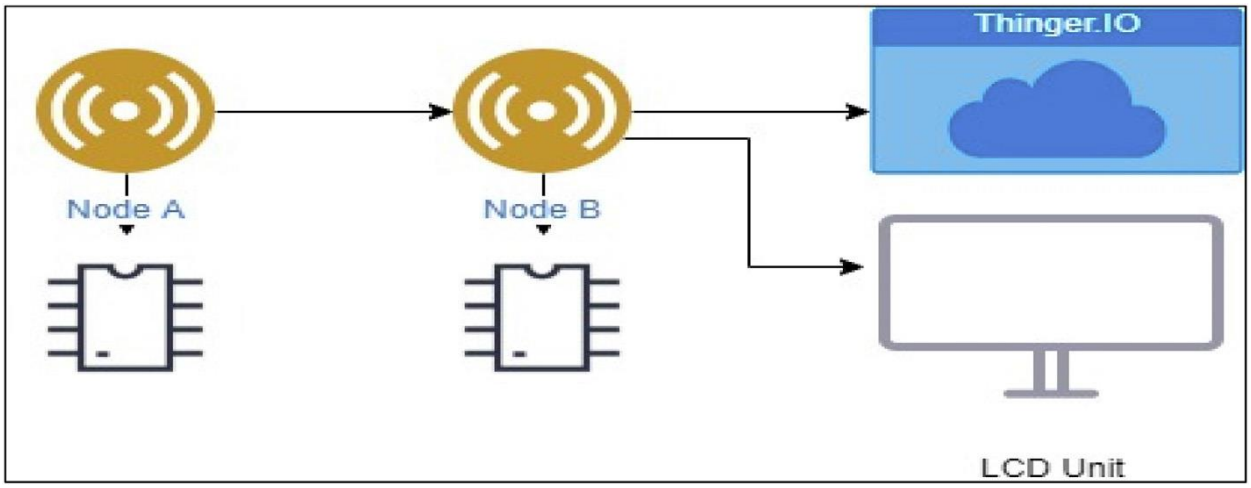
9. Visualization:

- Plot the data using appropriate visualization tools to identify patterns, outliers, or trends. Visualization can provide insights and guide further preprocessing or analysis steps.

Map processing evaluation:



Prototype communication:



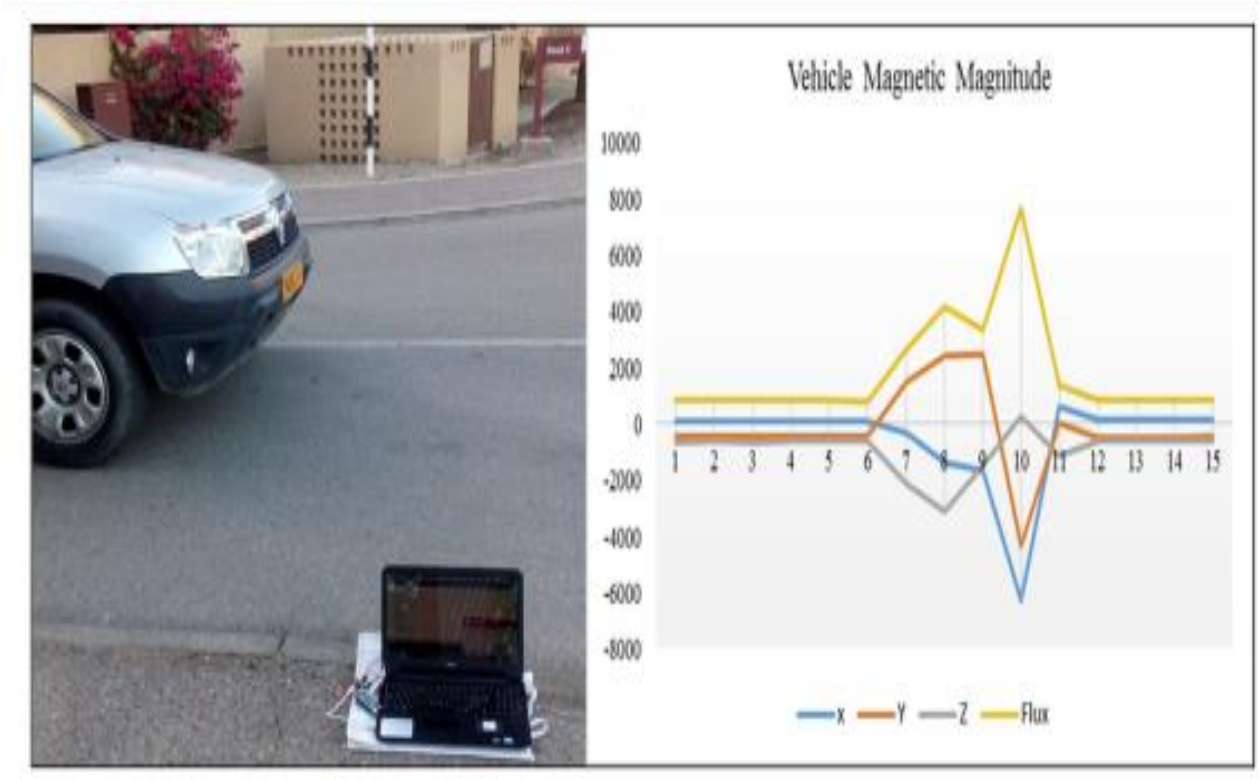
MongoDB and the pictorial view presented. The original map is on the left and expected junctions are marked on the right.

The accuracy of the map processing script is measured through precision and recall metrics. These metrics are used to evaluate the effectiveness of information retrieval.

Precision % ¼ (Number of relevant junctions retrieved/Number of junctions retrieved) *100% Recall %¼ (Number of relevant junctions retrieved/Number of relevant junctions in the map) *100%The script expects to retrieve the message board location according to the given threshold for the number of connected roads

Dataset (traffic_data.csv):

```
timestamp,  vehicle count, speed, vehicle type
2023-10-15 08:05:00, 12,        60,   Car
2023-10-15 08:10:00, 15,        55,   Car
2023-10-15 08:15:00, 10,        58,   Truck
```

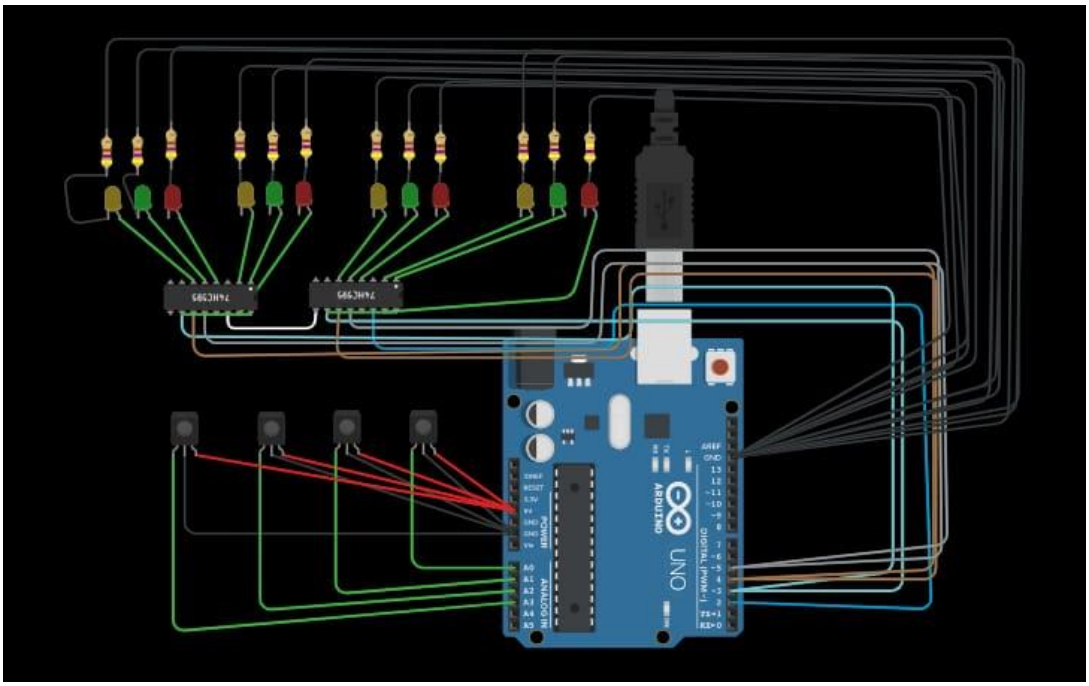


PYTHON SCRIPT:

```
import pandas as pd:
# Load Data
File path = 'traffic_data_sample.csv'
data = pd.read csv(file path)

# Data Cleaning
data['vehicle_count'].fillna(data['vehicle_count'].mean(), inplace=True)
data['speed'].fillna(data['speed'].mean(), inplace=True)
data.drop_duplicates(inplace=True)

# Date-Time Processing
data['timestamp'] = pd.to_datetime(data['timestamp'])
data['hour'] = data['timestamp'].dt.hour

# Data Filtering
data = data[(data['speed'] >= 0) & (data['speed'] <= 200)]

# Data Transformation - Categorization of speed
data['speed_category'] = pd.cut(data['speed'], bins=[0, 50, 100, 200], labels=['slow', 'medium', 'fast'])

# Data Aggregation
hourly_data = data.groupby('hour').agg({'speed': 'mean', 'vehicle_count': 'sum'}).reset_index()

# Feature Engineering - Traffic growth rate
hourly_data['growth_rate'] = hourly_data['vehicle_count'].diff().fillna(0)

# Store Processed Data
hourly_data.to_csv('processed_traffic_data.csv', index=False)
```

In this script:

- `timestamp` represents the time the data was recorded.
- `speed` indicates the average speed of the vehicles (e.g., in km/h).
- `vehicle_count` shows the number of vehicles that passed a specific point.
- `vehicle_type` could be categories like car, truck, bike, etc.



PROGRAM:

```
data_dict = {
   'timestamp': ['2023-10-15 08:05:00', '2023-10-15 08:10:00',
'2023-10-15 09:10:00', '2023-10-15 09:15:00'],
   'vehicle_count': [12, 15, 16, 18],
   'speed': [60, 45, 70, 85]
}
data = pd.DataFrame(data_dict)
Python Script for Processing the Assumed Data:
import pandas as pd
```

```
data_dict = {
   'timestamp': ['2023-10-15 08:05:00', '2023-10-15 08:10:00',
'2023-10-15 09:10:00', '2023-10-15 09:15:00'],
   'vehicle_count': [12, 15, 16, 18],
   'speed': [60, 45, 70, 85]
}
data = pd.DataFrame(data_dict)

# Date-Time Processing
data['timestamp'] = pd.to_datetime(data['timestamp'])
data['hour'] = data['timestamp'].dt.hour

# Data Filtering (speeds between 0 and 200 km/h)
data = data[(data['speed'] >= 0) & (data['speed'] <= 200)]

# Data Transformation - Categorizing speed
data['speed_category'] = pd.cut(data['speed'], bins=[0, 50, 100,
200], labels=['slow', 'medium', 'fast'])

# Data Aggregation
hourly_data = data.groupby('hour').agg({'speed': 'mean',
'vehicle_count': 'sum'}).reset_index()

# Feature Engineering - Traffic growth rate
hourly_data['growth_rate'] =
hourly_data['vehicle_count'].diff().fillna(0)

print(hourly_data)
```

Output:

	hour	speed	vehicle_count	growth_rate
0	8	52.5	27	0.0
1	9	77.5	34	7.0

Conclusion:

This research proposed an IoT based system model to collect, process, and store real-time traffic data. This research provided real-time traffic monitoring for traffic updates through roadside message units. The traffic authorities can also broadcast messages on VIP visits, medical emergencies, accidents, etc. to corresponding message units, which will assist the public in decision making and save their time on roads. The proposed system uses magnetic sensor nodes to collect real-time vehicle information. The real-time data is processed by WiFi-enabled microcontrollers and sends to an IoT platform for further actions