PUBLIC TRANSPORTATION OPTIMIZATION USING IOT

Member

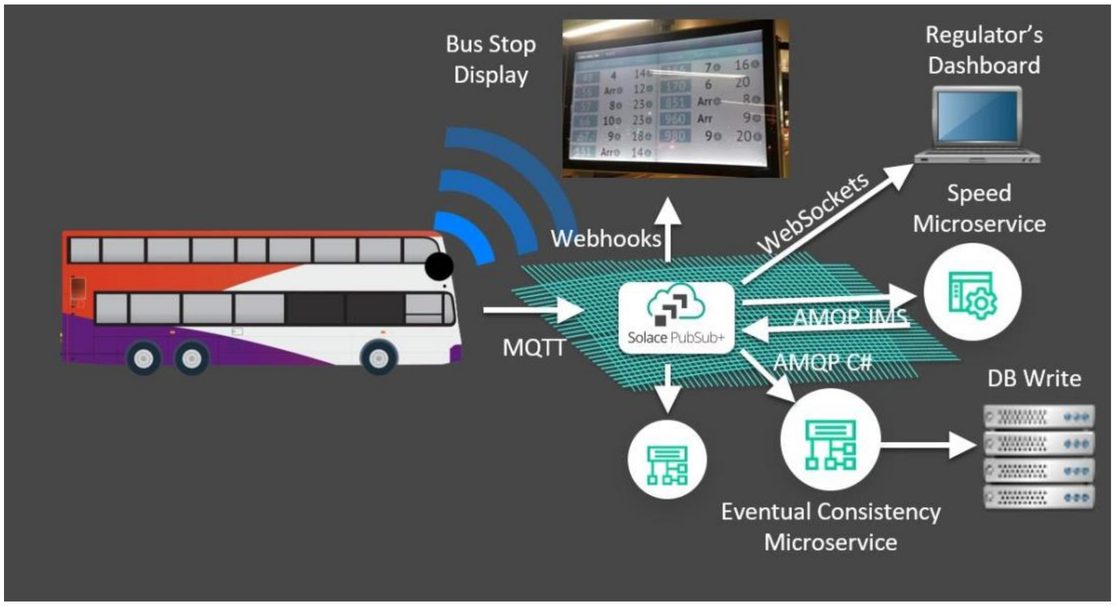
- Satheesh K

Phase 4 submission document

Project Title : PUBLIC TRANSPORTATION OPTIMIZATION

Phase 4 : Development Part 1

Topic : Start simulation of python script and algorithms for optimization of public transport.



Introduction

Optimizing the operations of public transport systems involves many levels, starting from strategic planning and moving to tactical planning and near real-time (operational) control. In a general planning process, we move from the Optimal Stop Location problem to Transit Network Design (TND), Frequency Setting (FS), Transit Network Timetabling (TNT), Vehicle Scheduling (VSP), and Crew Scheduling (CSP). This chapter covers the strategic public transport planning level, including aspects such as the estimation of trip distribution, the optimal stop location problem, routing problems, and the line planning problem that result in the design of the stations and the line routes of a transit network. The decisions about the locations of the stations and the routes of fixed line services are strategic because, once made, it is not easy to amend them within a short time.

Public transport optimization including sensors and components

In. Public ***transport optimization using*** Various sensors and components are used in public transport systems to enhance efficiency, safety, and passenger experience. Some of these include:

- GPS (Global Positioning System) Sensors
- Passenger Counting Sensors
- CCTV Cameras
- Ticketing and Fare Collection Systems
- Vehicle Health Monitoring Systems
- Automated Announcements and Information Displays
- Wi-Fi and Connectivity Components

1. GPS (Global Positioning System) Sensors: Used for real-time tracking and monitoring of vehicles to provide accurate location information.

2. Passenger Counting Sensors: Employed to measure the number of passengers boarding and alighting at different stops or stations, aiding in demand analysis and resource allocation.

3. CCTV Cameras: Installed for security purposes to ensure passenger safety and monitor any incidents that may occur on the vehicles or at the stations.

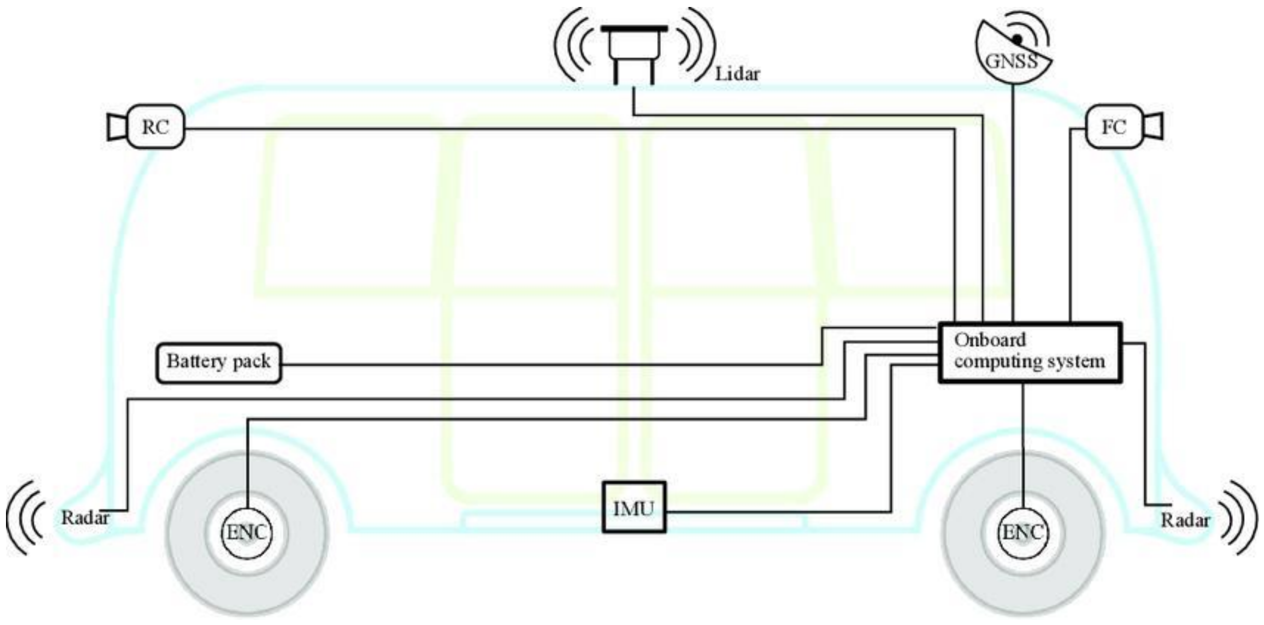
4. Ticketing and Fare Collection Systems: Components such as ticket machines, contactless smart cards, or mobile payment systems are utilized for convenient and efficient fare collection.

5. Vehicle Health Monitoring Systems: Sensors are employed to monitor the health and performance of the vehicles, including engine health, fuel efficiency, and maintenance requirements.

6. Automated Announcements and Information Displays: Components such as audio systems and digital displays are used to provide passengers with real-time information about routes, schedules, and any important updates.

7. Wi-Fi and Connectivity Components: Equipped to provide passengers with internet access and connectivity during their commute, enhancing the overall passenger experience.

These components collectively contribute to the effective operation, management, and safety of public transportation systems.



Optimizing public transportation using the Internet of Things (IoT) in a Python script is a complex task that involves collecting and analyzing real-time data from various sources such as sensors, GPS devices, and traffic cameras. Here's a simplified example of how you can start building a public transportation optimization system using Python and IoT concepts. Keep in mind that a complete system would require a lot more components and integration with existing transportation infrastructure.

1. Data Collection :

- Collect real-time data from various sources, such as GPS devices on vehicles, traffic cameras, weather sensors, and passenger counting sensors. You can use IoT protocols like MQTT or HTTP for data transmission.

2. Data Processing :

- Process the collected data to extract relevant information, such as vehicle locations, traffic conditions, and passenger counts.

3. Route Optimization:

- Use algorithms for route optimization to determine the most efficient routes for public transportation vehicles. You can use libraries like NetworkX for graph-based routing.

4.Traffic Monitoring :

- Analyze real-time traffic data to avoid congested routes and make on-the-fly adjustments to the planned routes.

5. Predictive Maintenance :

- Use IoT sensors on vehicles to monitor their health and predict maintenance needs, ensuring that vehicles are always in good working condition.

6. Passenger Information :

- Provide real-time information to passengers about the location of vehicles, estimated arrival times, and any delays. This can be done through a mobile app or information displays at stops.

In this simplified example, the script simulates the movement of public transportation vehicles and continuously updates their locations. You would need to integrate this with real data sources and more advanced algorithms for a complete system.

Keep in mind that building a production-ready public transportation optimization system is a complex task that requires careful planning, data integration, and the use of specialized libraries and tools.

Mini code:

```
import random

import time
```

```
# Simulated IoT data for vehicle locations
vehicle_data = {
    "bus_1": {"latitude": 40.7128, "longitude": -74.0060},
    "bus_2": {"latitude": 40.730610, "longitude": -73.935242},
    # Add more vehicles here
}

def simulate_vehicle_data():
    for vehicle_id, data in vehicle_data.items():
        # Simulate vehicle movement (randomly)
        data["latitude"] += random.uniform(-0.001, 0.001)
        data["longitude"] += random.uniform(-0.001, 0.001)
        print(f"{vehicle_id}: Latitude {data['latitude']}, Longitude
{data['longitude']}")

while True:
    simulate_vehicle_data()
    # Implement route optimization and traffic monitoring logic here
    time.sleep(10)  # Simulate data updates every 10 seconds
```

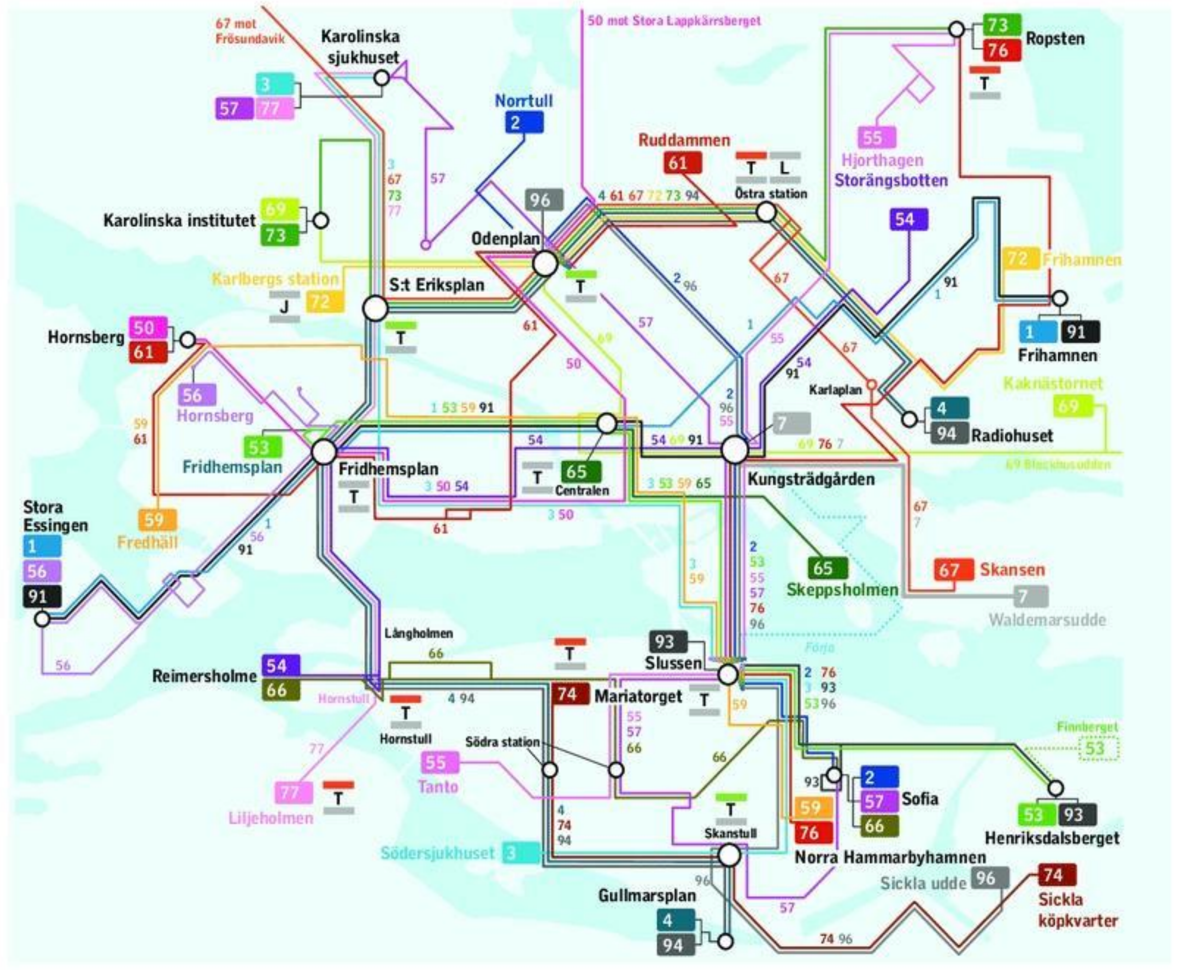
Python script for public transport systems

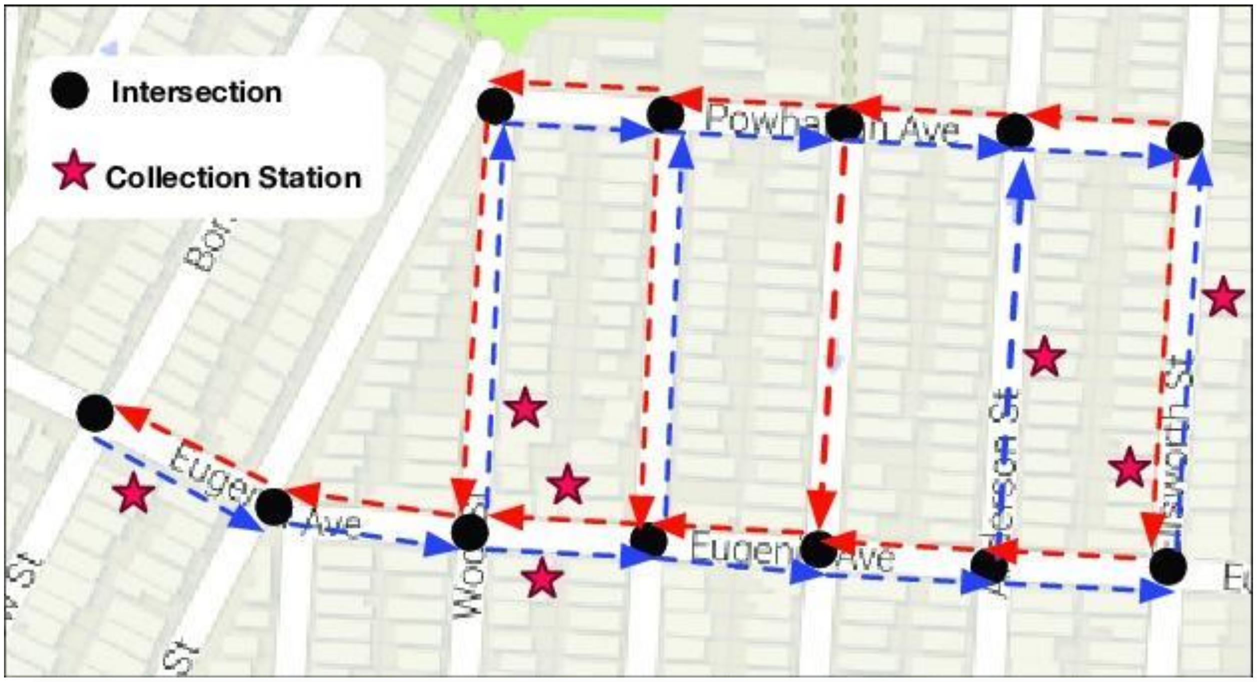
```
Import random

Import time

# Simulated IoT data for buses

Bus_data = [

    {“bus_id”: 1, “location”: (37.7749, -122.4194)},  # San Francisco

    {“bus_id”: 2, “location”: (34.0522, -118.2437)},  # Los Angeles

    # Add more buses and their locations

]

# Simulated passenger demand

Passenger_demand = [

    {“location”: (37.7749, -122.4194), “destination”: (34.0522, -
118.2437)},  # SF to LA

    # Add more passenger demand routes

]
```

```
Def optimize_routes(buses, demand):

   # Add your optimization logic here

   # For demonstration, we'll assign buses to the nearest passenger
demand

   Optimized_routes = []

   For passenger in demand:

      Min_distance = float("inf")

      Assigned_bus = None

      For bus in buses:

         Distance = haversine(bus["location"], passenger["location"])

         If distance < min_distance:

            Min_distance = distance

             Assigned_bus = bus["bus_id"]

      Optimized_routes.append({"passenger": passenger, "bus_id":
assigned_bus})
```

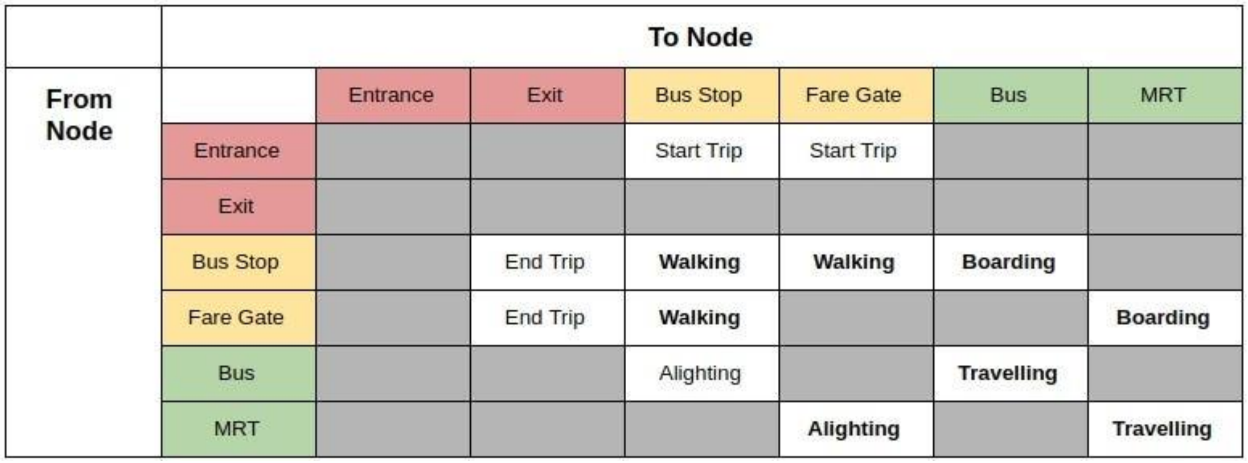
```
    Return optimized_routes

Def haversine(coord1, coord2):

    # Haversine formula to calculate distance between two coordinates

    Lat1, lon1 = coord1

    Lat2, lon2 = coord2

    # Calculate distance (for simplicity, this is not accurate for long
distances)

    Return random.uniform(10, 200)  # Simulated distance

While True:

    Optimized_routes = optimize_routes(bus_data, passenger_demand)

    For route in optimized_routes:

        Print(f"Bus {route['bus_id']} assigned to passenger route from
{route['passenger']['location']} to {route['passenger']['destination']}")

    Time.sleep(300)  # Simulated time delay, e.g., 5 mminutes
```

Output


67 mot Frösundavik
Karolinska sjukhuset
50 mot Stora Lappkärrsberget
Ropsten
Norrtull
Ruddammen
Östra station
Hjorthagen
Storängsbotten
Karolinska institutet
Odenplan
Frihamnen
Karlbergs station
S:t Eriksplan
Hornsberg
Hornsberg
Kaknästornet
Karlaplan
Radiohuset
Fridhemsplan
Fridhemsplan
Centralen
Kungsträdgården
69 Blockhusudden
Stora Essingen
Fredhäll
Skansen
Skeppsholmen
Waldemarsudde
Långholmen
Slussen
Färja
Reimersholme
Mariatorget
Hornstull
Södra station
Finnberget
Tanto
Sofia
Henriksdalsberget
Liljeholmen
Skanstull
Södersjukhuset
Norra Hammarbyhamnen
Sickla udde
Gullmarsplan
Sickla köpkvarter



<u>Node to Node Route Planning</u>



	To Node						
From Node		Entrance	Exit	Bus Stop	Fare Gate	Bus	MRT
	Entrance			Start Trip	Start Trip		
	Exit						
	Bus Stop		End Trip	**Walking**	**Walking**	**Boarding**	
	Fare Gate		End Trip	**Walking**			**Boarding**
	Bus			Alighting		**Travelling**	
	MRT				**Alighting**		**Travelling**

Conclusions

This paper introduced a novel IoT-based bus stop that provided smart monitoring and maintenance solutions to reduce energy consumption and increase the satisfaction of commuters. The system consisted of layers corresponding to data acquisition, processing, storage, and presentation. The system monitored the bus stop's occupancy and sensors so that based on the sensor readings, the lights and air-conditioning could operate more efficiently. The air pollution in and around the bus stop vicinity was also monitored. The bus stop operation status and air pollution levels were then sent to a cloud-based server. A mobile app was developed to enable operation engineers to monitor the air conditioning and lights remotely. Utilizing Google Maps, a bus stop operation status was displayed using different colored attributes. In order to meet the system objectives, test cases were conducted to ensure that the system performance was aligned with the goals. The results were conclusive to determine that this project can be scaled to multiple bus stops. The proposed system cuts down on power consumption by controlling the bus stop's utilities based on its occupancy. By keeping constant track of the bus stop's occupancy, the system can control the lights and the air conditioning. This helps save energy, as compared to traditional bus stops where the utilities remain running even during hours of no occupancy, leading to a waste of resources. The exact amount of saved energy will be reported in future work as we plan to add a solar energy system to supply power to the bus stop. Another aspect than can be addressed in the future is the communication between the vehicle and the infrastructure (Bus Stop). Additionally, the versatility of the system can also expand its application to various settings such as schools, weather stations, hospitals, and the like.