Public Transport Analysis Phase - 3

Development part-1

Hardware Setup:

Install IoT devices like GPS trackers, sensors, and cameras on public transportation vehicles (buses, trains, etc.).

Ensure these devices are capable of collecting data related to location, passenger count, weather conditions, and vehicle health.

Data Collection:

Create Python scripts to collect data from IoT devices. You may use libraries like pySerial or pyUSB to communicate with sensors and devices.

Store the collected data in a database (e.g., MySQL, MongoDB) or cloud storage.

Real-time Data Streaming:

Use IoT protocols like MQTT or WebSocket to stream real-time data from devices to a central server**.**

Route Optimization:

Implement route optimization algorithms to minimize travel time and energy consumption.

Use Python libraries like NetworkX or implement your custom routing algorithm.

Passenger Information:

Develop a mobile app or web interface for passengers to access real-time information about public transportation, including estimated arrival times and route information.

Use Python web frameworks like Django or Flask for the backend.

Data Visualization:

Create interactive data visualization using libraries like Matplotlib, Seaborn, or web-based tools like D3.js.

Visualize routes, vehicle locations, and passenger statistics.

Alerts and Notifications:

Implement alerting systems to inform passengers about delays, route changes, or other relevant information.

Use Python's notification libraries or messaging platforms.

Feedback and Reporting:

Gather feedback from passengers and transportation authorities.

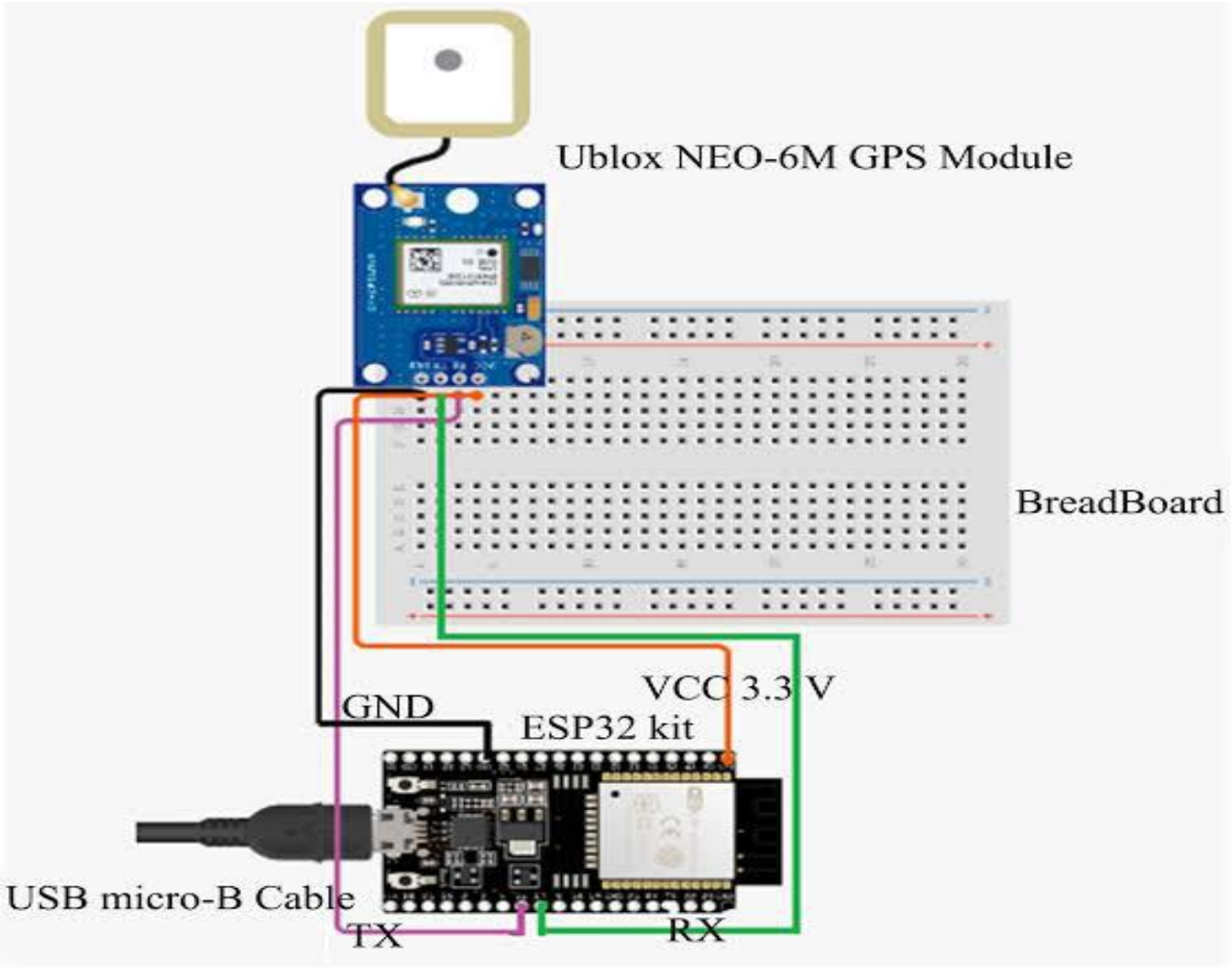
Generate reports and insights from the collected data for decision-making.

Security and Privacy:

Ensure the security of IoT devices and data by implementing encryption and access controls.

Testing and Deployment:

Thoroughly test the system in a controlled environment before deploying it in a real-world setting.



Maintenance:

Regularly update and maintain the system, including IoT device maintenance, software updates, and database management.

Please note that developing a complete public transportation optimization system using IoT is a substantial and complex project. You may need a team of experts in IoT, data science, and software development to accomplish this. Additionally, consider the legal and ethical aspects of collecting and using data from public transportation vehicles and passengers.

we'll focus on optimizing bus routes and providing real-time information to passengers.

PYTHON CODE:::

```
# Import necessary libraries (ensure you have these installed)
import random
import time
from threading import Thread

# Simulate IoT data from buses (longitude, latitude, passenger count)
class BusSimulator:
    def __init__(self, bus_id):
        self.bus_id = bus_id
        self.latitude = random.uniform(30.0, 35.0)
        self.longitude = random.uniform(-120.0, -115.0)
        self.passenger_count = random.randint(0, 50)

    def update_data(self):
        while True:
            # Simulate data changes
```

```
            self.latitude += random.uniform(-0.01, 0.01)
            self.longitude += random.uniform(-0.01, 0.01)
            self.passenger_count = random.randint(0, 50)
            time.sleep(10)  # Simulate data update every 10 seconds

# Simulate multiple buses
num_buses = 5
buses = []
for i in range(num_buses):
    bus = BusSimulator(bus_id=i)
    buses.append(bus)
    t = Thread(target=bus.update_data)
    t.daemon = True
    t.start()

# Passenger app to get real-time information
class PassengerApp:
    def __init__(self):
        pass

    def get_real_time_info(self, bus):
        while True:
            print(f"Bus {bus.bus_id} - Latitude: {bus.latitude:.4f}, Longitude: {bus.longitude:.4f}, Passengers: {bus.passenger_count}")
            time.sleep(10)  # Check for updates every 10 seconds

# Simulate passenger apps
```

```
passenger_apps = []
for bus in buses:
    passenger_app = PassengerApp()
    t = Thread(target=passenger_app.get_real_time_info, args=(bus,))
    t.daemon = True
    t.start()
```

****