NavAR: Implementation of a navigation system with ubiquitous computing techniques and calm technology

VICKY NASIELI, Computer Engineering and Informatics Department, UPatras, Greece
NIKOLAOS MAVRIAS, Computer Engineering and Informatics Department, UPatras, Greece

This paper shows the development of an information visualization system designed to assist drivers in making informed decisions on the road. Utilizing real-time road condition data, the system displays traffic levels on upcoming roads and provides alternative route suggestions to optimize travel time. The innovation is that the information is displayed through soft ubiquitous techniques either through a HUD, augmented reality glasses(AR) or the more futuristic augmeneted reality contact lenses. Our focus is on the development of a complete simulation, using real traffic data produced by a industry-wide accepted plugin and the connection of the data stream to a simulator where AR technology handle them to produce what would be displayed in a real scenario.

Additional Key Words and Phrases: traffic, AR, ML, uxsim, airsim, navigation, simulator, HUD

ACM Reference Format:
Vicky Nasieli and Nikolaos Mavrias. 2025. NavAR: Implementation of a navigation system with ubiquitous computing techniques and calm technology. 1, 1 (February 2025), 13 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 PROBLEM DESCRIPTION AND OBJECTIVES

We're steering into a meta-world, but we have not optimised solutions for the real world. In that matter, we found interest in two dimensions in the commute domain, first from the personal POV (Point-of-View) on the way the traffic information is being displayed with apps in our phones, and the dangers that this solution makes us suscept to when our point of focus is redirected from the road. Even so in situations where we travel solo, and there is no co-driver to take on navigation duties. With the second being in the collective POV of our society, from the traffic congestion that comes inevitably with the urbanization, where simple short-travels in cities become an anxiety-inducing experience and thus the daily life of the citizen has significant time wasted as well as the extra gas transmission that are emitted. Our objectives are per the first point set, that with the use of calm technologies the directions/info is displayed in a more non-obtrusive manner either in the HUD of the car or implemented with the use of AR in the form of smart glasses or contact lenses. Per the second point, the objective which we will not analyze further in this submission, would be the solution of the TSP especially with greedy algorithms.Those would significantly benefit from the real time info of the congestion status of each street, which then would be used to provide more accurate directions especially when stops are involved.

Authors' Contact Information: Vicky Nasieli, up1090034@upnet.gr, Computer Engineering and Informatics Department, UPatras, Patras, Greece; Nikolaos Mavrias, up1084664@upnet.gr, Computer Engineering and Informatics Department, UPatras, Patras, Greece.

Contents

2 ANALYSIS

2.1 Literature review

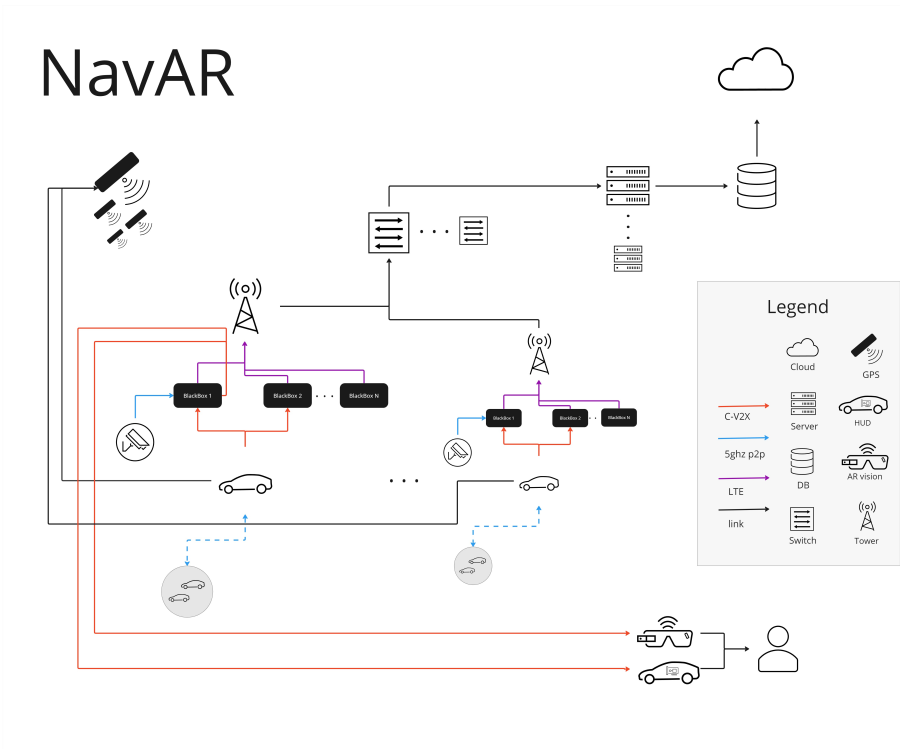
Integrating calm technology principles into AR and HUD systems for vehicle navigation helps minimize distractions while optimizing situational awareness.

Mark D. Weiser's concept of calm technology (1996) [2] aims to seamlessly integrate information into the user's environment without overwhelming their cognitive capacities, especially critical in driving where attention is paramount (Weiser, 1996).

Zhu Xuancheng et al. (2018) [3] developed an AR-HUD system utilizing GPS data to enhance navigation by correcting GPS inaccuracies due to latency and ensuring high real-time responsiveness, thereby minimizing the need for drivers to divert their attention from the road to find where they are be correcting themselves for the GPS delay. This approach aligns with calm technology by making navigation more intuitive and less intrusive (Zhu et al., 2018).

Additionally, research findings [1] indicated that well-designed AR-HUD interfaces, following human-computer interaction principles and visual design rules, can enhance cognitive resource allocation and driving safety. Furthermore, AR-HUD-assisted driving was observed to improve drivers' responses to unexpected driving scenarios based on the analysis of various driving performance indicators. Furthermore, it is imperative to consider suplementary domains of interest. As evidenced by the findings of (Hwang et al. 2016) [4] , it is indicated that the AR-HUD system could potentionally engender relaxation among drivers exhibiting high levels of interpersonal "In addition, it is imperative to consider supplementary domains of interest. The findings indicate that the Augmented Reality Head-Up Display (AR-HUD) system could potentially engender relaxation among drivers exhibiting high levels of interpersonal anxiety, while concurrently posing deleterious ramifications for individuals inclined towards sensation-seeking behavior. This observation underscores the likelihood of the system exerting dual influences on both visual safety-oriented driving conduct and the psychological aspect of driving safety."

2.2 NavAR System Architecture



The data acquired from the camera sensors, augmented by GPS information, is transmitted to a centralized processing unit, colloquially referred to as a "black box," where computational analyses are conducted. It is noteworthy that this processing unit is deployed at a frequency of one unit per every n blocks, ensuring efficient data aggregation and analytical processing across the monitored road network segments.

Subsequently, the processed data emanating from the black box undergoes transmission either to the Augmented Reality (AR) application, facilitated by the Cellular Vehicle-to-Everything (C-V2X) technology, or to the telecommunications tower utilizing Long-Term Evolution (LTE) infrastructure. Upon reaching the telecommunications tower, the data traverses through a network switch, which orchestrates its redirection to a designated server for subsequent processing stages. Following this initial processing phase, the data proceeds to a database repository, and subsequently, is channeled to cloud-based infrastructure for potential analytics processing and other pertinent operations. This sequential flow of data propagation ensures systematic handling and refinement of the collected traffic information, thereby optimizing its utility for various analytical endeavors and decision-making processes.

2.3 Choices made in the architecture

2.3.1 Hardware with ubiquitous computing.

Context Awareness. In line with Mark Weiser's concept of pervasive computing, integrating sensors into devices and vehicles enables the provision of contextually aware services that adjust to the user's surroundings and tasks

Energy Efficiency. With the use of devices for Low Power Wide Area Networks (LPWAN) we address the fundamental idea of ubiquitous computing—energy efficiency—by transmitting little amounts of data over great distances while preserving battery life.

Pervasiveness. Enhancing natural connection with technology through multi-modal interfaces (touch, speech, gesture) is crucial, according to Weiser, for technology to become ubiquitous but unobtrusive.

"Everyware". Implementing algorithms for wearable technology (for navigating pedestrian areas): This fits nicely with Adam Greenfield's Everyware theory, which holds that technology permeates every aspect of daily life.

Decentralization in Infrastracture. Stressing decentralised computing, which uses bandwidth-saving data processing at the network's edge to cut down on latency and usage of bandwidth—two factors, are vital to ubiquitous computing for real-time applications.

Connectivity. With the use of Sturdy Antenna Systems for C-V2X, enduring connectivity is provided, which is essential to ubiquitous systems and guarantees continuous communication even in fast-paced urban areas.

2.3.2 Software with ubiquitous computing.

Context-aware systems. Referring to software that adapts its operations due to changes in the environment, creating extra computing layers that respond to the context. In our case adjusting the navigation routes based on traffic congestion in real time.

Decentralization in data processing algorithms. Data computations are done in the place they are collected, giving scalability and responsiveness in ubiquitous systems. For us meaning in the car level, at worst in the black-box level and not higher.

Universal design. Software that adapts based on user preferences and abilities, regardless of their technical skills and/or disabilities, making the technology accessible to all. The main reason NavAR is proposed with its output being either thru HUD or glasses-lens.

3 OUR PROPOSED APPROACH

The proposed complete system called NavAR, which architecture was analyzed above, is obviously an industry wide proposition to be applied. However we will recreate a minimal real scenario with the use of a data producing script, in Python, widely used in the academic domain called UXSim for the production of congestion data along with the use of a Microsoft-backed plugin for simulation of the AR notifications/navigation, called AirSim, running inside the well-established Unity with the simSetPose() API. The basic idea is to extract data from the traffic nodes and analyze them to gain insights for traffic congestion and then visualize the produced directions with the use of calm technology.

4 TECHNICAL ANALYSIS - CODE IMPLEMENTATION

4.1 Our approach

The basic implementation idea is to utilize the data generated by the uxsim simulator , essentially replacing for the purposes of this work, the sensors and the real road network so we can present our proposed navigation system in Unity. In respect to the flow of data they are passed through a number of different technologies, each contributing with a different role. For name's shake those are UXSim, Apache's Zookeeper, Kafka(producer-consumer), a localhost flask server and lastly arriving in Unity where our unobtrustive navigation system is presented as a simulation.

4.2 Reviewing in - In person and UTube link

Now we do know that going through the proper installation route following this section is kind of troublesome, even we in each new installation find bugs. So we have setted it up and running, in a computer of the USIDAS lab at the University of Patras, Greece where you may interact with the game in the push of just 4 buttons, 5 if you count powering on the PC. If you're not in the greater area of Patras, rest assured we have set up a anydesk so contact us to arrange the credentials for a remote session.
Lastly you may find a gameplay recorded by us, demonstrating the navigation system in the following YouTube link

4.3 Installation - Run Instructions

Apache Kafka Installation - Steps for Windows Installation

1. Download Kafka from the official Apache Kafka website: Kafka Installation

• Scala 2.13 - kafka_2.13-3.8.0.tgz (asc, sha512)

System Requirements:

- Java JDK 17 (ensure it is added to the system PATH)
- Windows Subsystem for Linux (WSL)

Kafka Installation Directory: `C:\`

After installation, edit the following files:

- `...\kafka\config\server.properties`
  Uncomment the following lines:

  ```
  listeners = PLAINTEXT://your.host.name:9092
  advertised.listeners = PLAINTEXT://localhost:9092
  ```

  Set `your.host.name` to `localhost`. Also, set the log directory:

  ```
  log.dirs=c:/kafka/kafka-logs
  ```

- `...\kafka\config\zookeeper.properties`
  Set the data directory:

  ```
  dataDir=c:/kafka/zookeeper-data
  ```

Running the Kafka Server: *Each command must be executed in seperate cmd terminals.*

- Start Zookeeper:

```
> bin\windows\zookeeper-server-start.bat config\zookeeper.properties
```

- Start Kafka server:

```
> bin\windows\kafka-server-start.bat config\server.properties
```

- Create a topic:

```
> bin\windows\kafka-topics.bat --create --bootstrap-server localhost:9092 --replication-factor 1
 --partitions 1 --topic simulation_results
```

- Start Consumer:

```
> bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic simulation_results
 --from-beginning
```

Create a new folder named “Scripts” in the directory: `C:\kafka\`
Save the provided file (.zip) to the "Scripts" folder. (e.g., `send_json_to_kafka.bat`)

5 CODE ANALYSIS

Initially, two distinct approaches were considered for testing. The first approach involves tracking the vehicle via its unique `vehicle_id` and calculating road traffic whenever the vehicle changes links. Road traffic is determined by first identifying the neighboring links and recording their demand, average travel time, and number of vehicles ahead of the user. This data is then compared to the traffic on the current link or defined route. If the traffic on the current route exceeds that of the neighboring links, the user is prompted with a recommendation to turn left or right to avoid congestion. Conversely, if the traffic on the current route is lower than that of the neighboring links, the user is advised to continue along the predefined route.

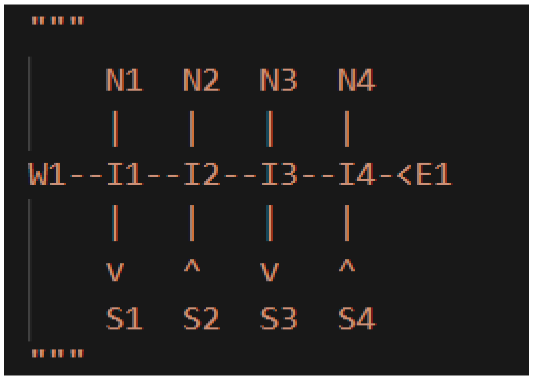
The second more advanced approach involves implementing a continuous traffic monitoring system across all nodes and links within the network. In this approach, traffic data is collected in real-time at every intersection and along each road segment. By employing algorithms to analyze traffic conditions throughout the entire route, the system can provide dynamic route suggestions based on current traffic levels. If a significant delay is detected on the planned route, the system proactively suggests alternative routes, taking into account factors such as traffic congestion, average travel times, and user preferences. This holistic view of traffic conditions ensures that the user receives the most efficient routing advice at all times.

5.1 UXSim part

. In this application, we make full use of the `uxsim` library, which is proposed for the implementation of this particular assignment. Our basic approach follows this flow:

The `uxsim` network performs a simulation (here, of one hour duration) and generates data.

For the simulation, the following schema is used as a road network representation (as defined in the given example file):



```
"""
    N1  N2  N3  N4
    |   |   |   |
W1--I1--I2--I3--I4-<E1
    |   |   |   |
    v   ^   v   ^
    S1  S2  S3  S4
"""
```

A `World` object instance is created with relevant parameters (see `uxsim` documentation).

5.1.1 Vehicle and Link Data. In addition, we define a `links` list and a `vehicle_ids` dictionary, which store the names of the created links and vehicle IDs, respectively.

To add the vehicles, the `World`'s function `addVehicle()` is used. We keep the vehicle IDs in a dictionary so that we can "follow" a certain car's path. The vehicle itself represents the user within the road network. We thought this was necessary because when we set the platoon variable value to "1", the simulation took an enormously long time executing, draining our system resources.

5.1.2 Tracking Vehicle Routes. `traveled_route(s)`: This function, already defined in the `uxsim` source code, had to be slightly modified. It captures the route taken by a vehicle by checking the vehicle's log of links and timestamps to create a route history, which will be used later. The parameter `s` represents a `World` class instance.

5.1.3 Other Key Functions.

- `print_traveled_route(vehicle, world)`: Prints the traveled route of the vehicle and checks traffic conditions before moving to each link.
- `capture_simulation_data(s, vehicle, next_link)`: Captures relevant simulation data and issues warnings if conditions exceed thresholds.
- `get_neighboring_links(link)`: Returns a list of neighboring links for the given link.
- `get_traffic_volume(link)`:Returns the traffic volume for the given link.

5.1.4 Extracting Network Data. The function `capture_simulation_data(s, vehicle, next_link)` is crucial for building and populating a JSON file with pertinent traffic data. This function collects link-level analysis results and determines whether the traffic conditions for the next link exceed predefined thresholds for traffic volume, remaining vehicles, and average journey time. When any of these circumstances are met, the `issue_warning` function is called to generate a thorough warning that includes important information including the vehicle's ID, current and next connections, traffic congestion, and options for alternate routes. The resulting warning data is formatted as a dictionary and written to a JSON file called `simulation_data.json` using the `write_warning_to_file(warning_data)` function.

5.1.5 Calculating Traffic Threshold. There are two ways to calculate the traffic threshold. The first one is to parse all traffic values from each link in the network, perform a median/average computation and decide on the threshold based on the standard deviation formula. The second approach, which was used in this implementation, is to manually set thresholds with static values.

5.1.6 Producing a Turn: Left/Right, Ahead Decision. We came up with two different ways for this decision. Due to time limitations, we opted for the more straightforward solution: mapping the links to their corresponding turn directions. This approach simplifies the routing logic by associating each link with a specific directional instruction. The only constraint here, is that the user must move from node E1 to W1 (As tested for vehicle with vehicle_id = 5)
The second approach builds upon the first by incorporating an additional layer of logic. For this method, we need to examine the vehicle's origin route to determine its starting point. By identifying the starting node, we can create a mapping for the opposite direction. This allows the system to provide routing instructions not only for forward movement but also for returning to the origin. Thus, the vehicle can effectively navigate both to and from the designated nodes, enhancing the overall flexibility and usability of the routing system.

5.2 Kafka part

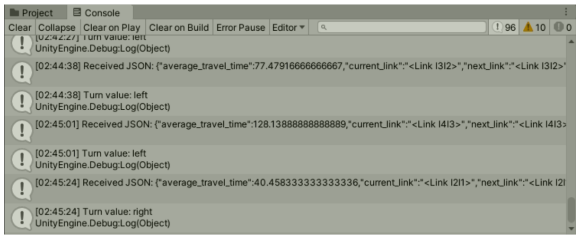
Apache Kafka represents a robust solution for the implementation of an advanced traffic monitoring and management system, owing to its high throughput, fault tolerance, and inherent scalability. The distributed architecture of Kafka enables the efficient handling of large volumes of real-time traffic data generated by vehicles and infrastructure sensors. This capability to process streaming data facilitates continuous analysis and allows for rapid responses to dynamic traffic conditions. Furthermore, Kafka's publish-subscribe model enhances communication between various system components, ensuring timely updates and alerts for end users. Consequently, drivers are equipped with accurate and current information regarding traffic patterns, thereby enabling informed decision-making regarding their routes. Additionally, Kafka's integration capabilities with other data processing frameworks augment the overall functionality of the traffic management system, allowing for sophisticated analytics and machine learning applications aimed at optimizing traffic flow and mitigating congestion.This multifaceted approach positions Apache Kafka as an invaluable asset in the pursuit of efficient and effective traffic management solutions, particularly through its support for decentralization—a key factor in ubiquitous computing. By enabling distributed data processing and real-time communication across multiple nodes, Kafka empowers various components of the traffic management system to operate independently while still maintaining cohesion.

5.3 Flask part

We chose a Flask server to pass data to Unity, as we can easily replace it with proper API calls to have a simulation with real data and sensors. Now it uses web requests to fetch pre-exported json messages from kafka consumer, as we sadly didn't manage despite our best efforts to establish a live connection between the two. It refreshes every 10 seconds which is also not ideal, but for the sake of our simulation is pretty acceptable. It is initialized with the code in the correspondingly named jupyter notebook, in the zip provided.

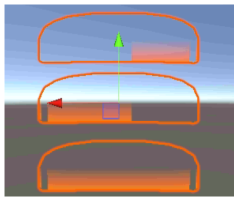
5.4 Unity part

5.4.1 ***Fetching and processing data - Csharp.*** The Unity script, HUDController, is responsible for controlling the heads-up display of a car navigation system by fetching traffic data from the aforementioned Flask server. The script uses UnityWebRequest to send HTTP GET requests to a server and retrieve JSON navigation data, which is then parsed using JsonUtility. Based on the "turn" value in the received data, the script activates one of three HUD elements implemented as gradiented windshields overlayed on top of one another: HUD-AHEAD, HUD-LEFT, or HUD-RIGHT, indicating the suggested turn direction. The HUDControlRoutine coroutine manages this behavior in a loop. Initially, the HUD displays "ahead" for 10 seconds before checking if a left or right turn is suggested, in which case it activates the respective HUD for 5 seconds. Afterward, the cycle repeats. The FetchNavigationData coroutine handles the server communication, retrieves the JSON response, and updates the lastTurn variable accordingly. The SetActiveHUD function controls which HUD element is active based on the direction provided.



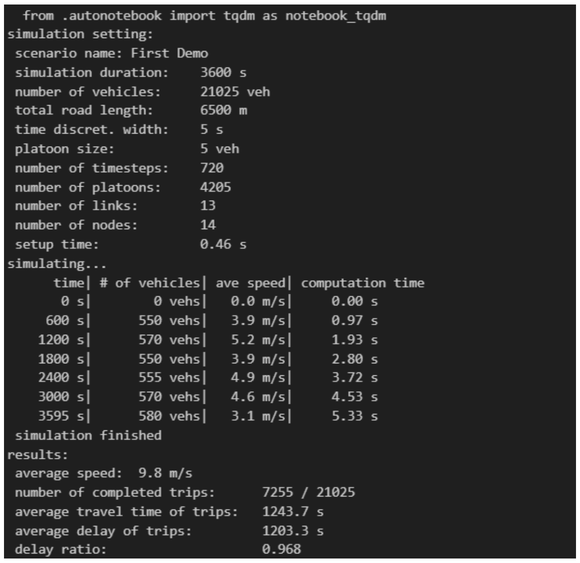
```
[02:42:27] Turn value: left
UnityEngine.Debug:Log(Object)
[02:44:38] Received JSON: {"average_travel_time":77.47916666666667,"current_link":"<Link I3I2>","next_link":"<Link I3I2>'
[02:44:38] Turn value: left
UnityEngine.Debug:Log(Object)
[02:45:01] Received JSON: {"average_travel_time":128.13888888888889,"current_link":"<Link I4I3>","next_link":"<Link I4I3>
[02:45:01] Turn value: left
UnityEngine.Debug:Log(Object)
[02:45:24] Received JSON: {"average_travel_time":40.458333333333336,"current_link":"<Link I2I1>","next_link":"<Link I2I
[02:45:24] Turn value: right
UnityEngine.Debug:Log(Object)
```

5.4.2 ***Custom shader generation code - HLSL.*** We used the High-Level-Shading-Language to generate the gradients needed. This choice was made considering it's applications on premier software like DirectX and considering that the generated gradients are responsive in real time so it's a perfect match for Unity. The generated shader creates a smooth gradient effect from opaque to transparent with the colour chosen being orange. The colour isn't random as the rather obvious choice being green could easily be minterpretated as go in a green light and cause accidents. Thus considering that and the colour's blending abilities in daytime as well as nightime it was the right choice. There are 3 different shader scripts, each to be applied to a different case. Only substantial differences between them are the coordinates compliments (u=1-v and u=u) in the cases of Left and Right direction as well as the orientation change in the case of Ahead direction.Then each shader is applied to a Material in Unity, which takes it's gradient properties. Lastly that custom material is applied to duplicates of the already provided windshield element which said windshields are then overlayed hidden atop each other. The scripts use properties like -Color and -Transparency to control the color and transparency transitions, while -GradientHeight determines the vertical range of the gradient.



6 TEST RUN

Each piece of code in the notebook is systematically numbered to enhance clarity and organization, with the format ’PART 1’, ’PART 2’, ..., ’PART N’. Running the first part on the notebook, produces the world (W) instance and performs its initialization:

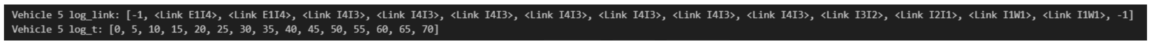


```
  from .autonotebook import tqdm as notebook_tqdm
simulation setting:
 scenario name: First Demo
 simulation duration:    3600 s
 number of vehicles:     21025 veh
 total road length:      6500 m
 time discret. width:    5 s
 platoon size:           5 veh
 number of timesteps:    720
 number of platoons:     4205
 number of links:        13
 number of nodes:        14
 setup time:             0.46 s
simulating...
      time| # of vehicles| ave speed| computation time
       0 s|        0 vehs|   0.0 m/s|     0.00 s
     600 s|      550 vehs|   3.9 m/s|     0.97 s
    1200 s|      570 vehs|   5.2 m/s|     1.93 s
    1800 s|      550 vehs|   3.9 m/s|     2.80 s
    2400 s|      555 vehs|   4.9 m/s|     3.72 s
    3000 s|      570 vehs|   4.6 m/s|     4.53 s
    3595 s|      580 vehs|   3.1 m/s|     5.33 s
 simulation finished
results:
 average speed:  9.8 m/s
 number of completed trips:      7255 / 21025
 average travel time of trips:   1243.7 s
 average delay of trips:         1203.3 s
 delay ratio:                    0.968
```

The next part (PART 2), compiles the following functions and produces no output:

```
traveled_route(s)
print_traveled_route(vehicle, world)
```

PART 3 , is showing the route for a predefined vehicle. (Here: Vehicle with vehicle_id = 5). The value "-1" suggests that the particular vehicle is out of the network, in this case, before it enters, and when it exits/finishes its route.



```
Vehicle 5 log_link: [-1, <Link E1I4>, <Link E1I4>, <Link I4I3>, <Link I4I3>, <Link I4I3>, <Link I4I3>, <Link I4I3>, <Link I4I3>, <Link I4I3>, <Link I3I2>, <Link I2I1>, <Link I1W1>, <Link I1W1>, -1]
Vehicle 5 log_t: [0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70]
```

PART 4 sets static thresholds which are used for computing traffic volumes.

```
TRAFFIC_VOLUME_THRESHOLD = 500  # Example threshold for traffic volume
VEHICLES_REMAIN_THRESHOLD = 10   # Example threshold for vehicles remaining
AVERAGE_TRAVEL_TIME_THRESHOLD = 30  # Example threshold for average travel time
```

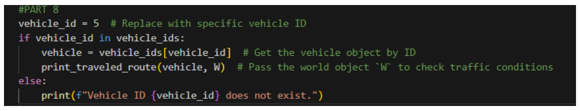
PART 6 compiles following functions and produces no output:

```
capture_simulation_data(s, vehicle, next_link)
issue_warning(vehicle_name, link, traffic_volume, vehicles_remain, average_travel_time,next_link)
write_warning_to_file(warning_data)
```

PART 7 compiles following functions and produces no output:

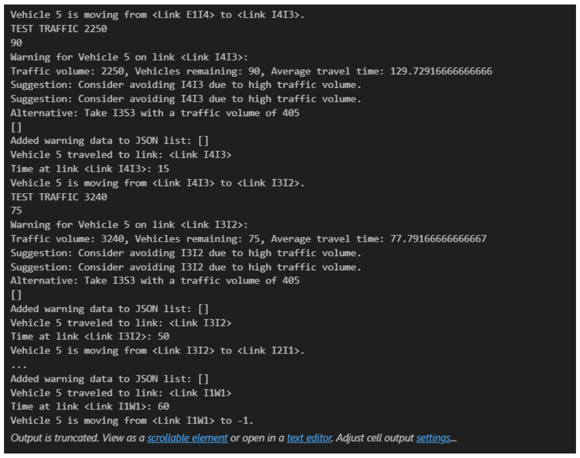
```
get_neighboring_links(link)
get_traffic_volume(link)
```

PART 8 is the final part. We must set a vehicle_id value to track its course.



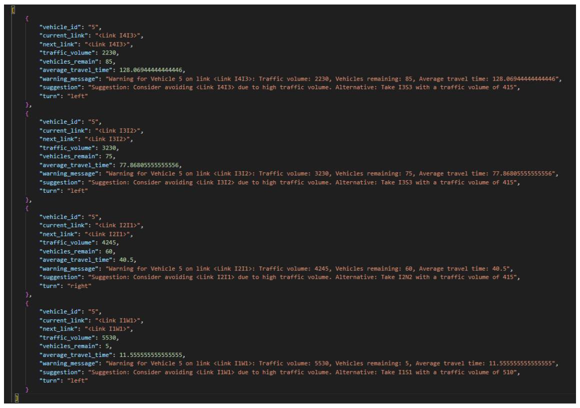
```
#PART 8
vehicle_id = 5  # Replace with specific vehicle ID
if vehicle_id in vehicle_ids:
    vehicle = vehicle_ids[vehicle_id]  # Get the vehicle object by ID
    print_traveled_route(vehicle, W)  # Pass the world object `W` to check traffic conditions
else:
    print(f"Vehicle ID {vehicle_id} does not exist.")
```

The final results are produced,the output presents the results of a traffic simulation run, which lasted for 3600 seconds and involved 21,025 vehicles across a 6,500-meter road network. Key metrics include an average speed of 9.8 m/s, with 7,255 vehicles completing their trips, and an average travel time of 1,243.7 seconds, indicating significant delays. Notably, the output includes warnings for Vehicle 5, detailing high traffic volumes and suggesting alternative routes to improve travel efficiency. Additionally, the simulation logs the path taken by Vehicle 5 and writes warning data to a JSON file for further analysis.



```
Vehicle 5 is moving from <Link E1I4> to <Link I4I3>.
TEST TRAFFIC 2250
90
Warning for Vehicle 5 on link <Link I4I3>:
Traffic volume: 2250, Vehicles remaining: 90, Average travel time: 129.72916666666666
Suggestion: Consider avoiding I4I3 due to high traffic volume.
Suggestion: Consider avoiding I4I3 due to high traffic volume.
Alternative: Take I3S3 with a traffic volume of 405
[]
Added warning data to JSON list: []
Vehicle 5 traveled to link: <Link I4I3>
Time at link <Link I4I3>: 15
Vehicle 5 is moving from <Link I4I3> to <Link I3I2>.
TEST TRAFFIC 3240
75
Warning for Vehicle 5 on link <Link I3I2>:
Traffic volume: 3240, Vehicles remaining: 75, Average travel time: 77.79166666666667
Suggestion: Consider avoiding I3I2 due to high traffic volume.
Suggestion: Consider avoiding I3I2 due to high traffic volume.
Alternative: Take I3S3 with a traffic volume of 405
[]
Added warning data to JSON list: []
Vehicle 5 traveled to link: <Link I3I2>
Time at link <Link I3I2>: 50
Vehicle 5 is moving from <Link I3I2> to <Link I2I1>.
...
Added warning data to JSON list: []
Vehicle 5 traveled to link: <Link I1W1>
Time at link <Link I1W1>: 60
Vehicle 5 is moving from <Link I1W1> to -1.
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

We can now check if the json output is geenrated and if the instructions specified are correct:

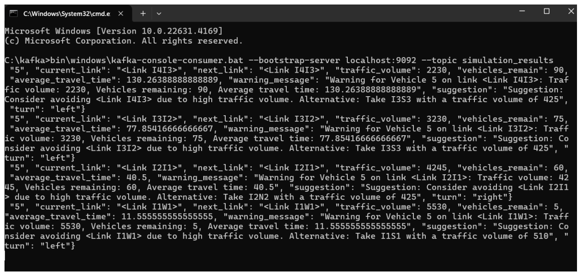


```
[
    {
        "vehicle_id": "5",
        "current_link": "<Link I4I3>",
        "next_link": "<Link I4I3>",
        "traffic_volume": 2230,
        "vehicles_remain": 85,
        "average_travel_time": 128.06944444444446,
        "warning_message": "Warning for Vehicle 5 on link <Link I4I3>: Traffic volume: 2230, Vehicles remaining: 85, Average travel time: 128.06944444444446",
        "suggestion": "Suggestion: Consider avoiding <Link I4I3> due to high traffic volume. Alternative: Take I3S3 with a traffic volume of 415",
        "turn": "left"
    },
    {
        "vehicle_id": "5",
        "current_link": "<Link I3I2>",
        "next_link": "<Link I3I2>",
        "traffic_volume": 3230,
        "vehicles_remain": 75,
        "average_travel_time": 77.86805555555556,
        "warning_message": "Warning for Vehicle 5 on link <Link I3I2>: Traffic volume: 3230, Vehicles remaining: 75, Average travel time: 77.86805555555556",
        "suggestion": "Suggestion: Consider avoiding <Link I3I2> due to high traffic volume. Alternative: Take I3S3 with a traffic volume of 415",
        "turn": "left"
    },
    {
        "vehicle_id": "5",
        "current_link": "<Link I2I1>",
        "next_link": "<Link I2I1>",
        "traffic_volume": 4245,
        "vehicles_remain": 60,
        "average_travel_time": 40.5,
        "warning_message": "Warning for Vehicle 5 on link <Link I2I1>: Traffic volume: 4245, Vehicles remaining: 60, Average travel time: 40.5",
        "suggestion": "Suggestion: Consider avoiding <Link I2I1> due to high traffic volume. Alternative: Take I2N2 with a traffic volume of 415",
        "turn": "right"
    },
    {
        "vehicle_id": "5",
        "current_link": "<Link I1W1>",
        "next_link": "<Link I1W1>",
        "traffic_volume": 5530,
        "vehicles_remain": 5,
        "average_travel_time": 11.555555555555555,
        "warning_message": "Warning for Vehicle 5 on link <Link I1W1>: Traffic volume: 5530, Vehicles remaining: 5, Average travel time: 11.555555555555555",
        "suggestion": "Suggestion: Consider avoiding <Link I1W1> due to high traffic volume. Alternative: Take I1S1 with a traffic volume of 510",
        "turn": "left"
    }
]
```

Vehicle with id=5, is on link I4I3 (Traffic volume 2230) and the system suggests to take a left turn on I3S3 (Traffic volume 425), which is the correct road direction, so our system works. The suggestions are made each time the user changes a link, up until the vehicle exits the network.

We then start the zookeeper service and the kafka server. Next, we must enable the consumer for the topic "simulation_results". Lastly, we must run yet another cmd terminal and execute the send_data_to_json.bat file.

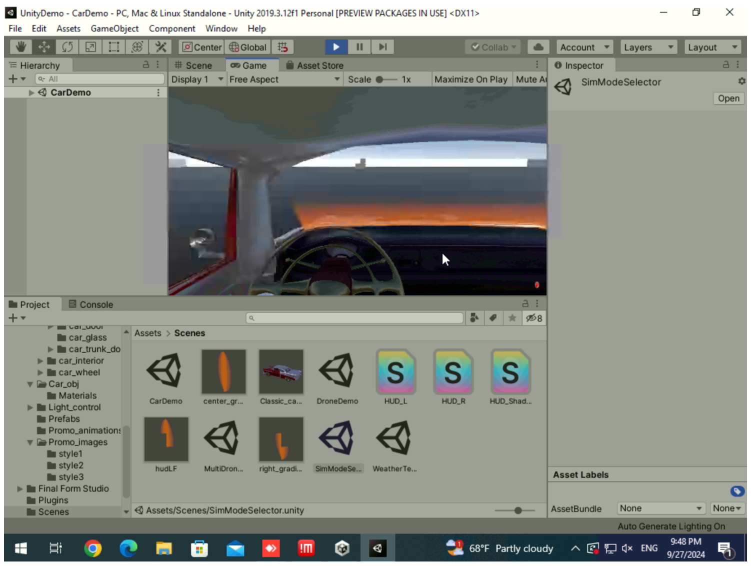
If the file is run succesfully, the kafka consumer must be populated with the json data, like in this case:



```
Microsoft Windows [Version 10.0.22631.4169]
(c) Microsoft Corporation. All rights reserved.

C:\kafka>bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic simulation_results
 "5", "current_link": "<Link I4I3>", "next_link": "<Link I4I3>", "traffic_volume": 2230, "vehicles_remain": 90,
 "average_travel_time": 130.26388888888889, "warning_message": "Warning for Vehicle 5 on link <Link I4I3>: Traf
fic volume: 2230, Vehicles remaining: 90, Average travel time: 130.26388888888889", "suggestion": "Suggestion:
Consider avoiding <Link I4I3> due to high traffic volume. Alternative: Take I3S3 with a traffic volume of 425",
 "turn": "left"}
 "5", "current_link": "<Link I3I2>", "next_link": "<Link I3I2>", "traffic_volume": 3230, "vehicles_remain": 75,
 "average_travel_time": 77.85416666666667, "warning_message": "Warning for Vehicle 5 on link <Link I3I2>: Traff
ic volume: 3230, Vehicles remaining: 75, Average travel time: 77.85416666666667", "suggestion": "Suggestion: Co
nsider avoiding <Link I3I2> due to high traffic volume. Alternative: Take I3S3 with a traffic volume of 425", "
turn": "left"}
 "5", "current_link": "<Link I2I1>", "next_link": "<Link I2I1>", "traffic_volume": 4245, "vehicles_remain": 60,
 "average_travel_time": 40.5, "warning_message": "Warning for Vehicle 5 on link <Link I2I1>: Traffic volume: 42
45, Vehicles remaining: 60, Average travel time: 40.5", "suggestion": "Suggestion: Consider avoiding <Link I2I1
> due to high traffic volume. Alternative: Take I2N2 with a traffic volume of 425", "turn": "right"}
 "5", "current_link": "<Link I1W1>", "next_link": "<Link I1W1>", "traffic_volume": 5530, "vehicles_remain": 5,
"average_travel_time": 11.555555555555555, "warning_message": "Warning for Vehicle 5 on link <Link I1W1>: Traff
ic volume: 5530, Vehicles remaining: 5, Average travel time: 11.555555555555555", "suggestion": "Suggestion: Co
nsider avoiding <Link I1W1> due to high traffic volume. Alternative: Take I1S1 with a traffic volume of 510", "
turn": "left"}
```

The data is subsequently processed in the Flask-Unity section of the paper. And the working result is displayed as in the screenshot below, we do strongly recommend you see the YouTube video in the 4th section or even experiment yourself with the game.


UnityDemo - CarDemo - PC, Mac & Linux Standalone - Unity 2019.3.12f1 Personal [PREVIEW PACKAGES IN USE] <DX11>
File Edit Assets GameObject Component Window Help
Hierarchy
CarDemo
Scene Game Asset Store
Display 1 Free Aspect Scale 1x Maximize On Play
Inspector
SimModeSelector
Open
Project Console
Assets > Scenes
CarDemo
DroneDemo
HUD_L
HUD_R
hudLF
SimModeSe...
WeatherTe...
Assets/Scenes/SimModeSelector.unity
Asset Labels
AssetBundle None None
Auto Generate Lighting On
68°F Partly cloudy
9:48 PM
9/27/2024

7 SUGGESTIONS FOR IMPROVEMENT AND FUTURE EXTENSIONS

I. **Implementation of set_route() Functionality:** Making use of the already defined set-_route() function would allow users to manually define a vehicle's route, offering greater control over the navigation process. This feature would enable the system to redefine any changed links in response to real-time traffic conditions. By allowing users to specify preferred routes, the system can enhance its adaptability, offering directions for longer routes.

II. **Integration of AI and Machine Learning:** The incorporation of artificial intelligence and machine learning algorithms could significantly enhance the traffic management system's predictive capabilities. By analyzing historical traffic data and patterns, the system can learn and anticipate congestion points, enabling it to make more informed routing suggestions. Machine learning models could also be employed to optimize traffic signal timings and improve overall system efficiency, leading to reduced travel times and increased user satisfaction.A promising approach is the use of federated learning, which allows multiple devices to collaboratively train machine learning models without sharing raw data.

III. **Testing with Real Users:** Conducting testing with real users is essential for evaluating the user experience (UX) of the traffic management application. Engaging with users in a real-world context allows for the identification of usability issues and areas for enhancement. User feedback can provide valuable insights into the effectiveness of the system's features, helping to refine the interface and overall functionality.

REFERENCES

[1] Anonymous. 2022. Does Augmented Reality Head-Up Display Help? A Preliminary Study. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (2022).

[2] Mark D. Weiser. 1996. Designing Calm Technology. *PowerGrid Journal* 1.01 (1996).

[3] Zhu Xuancheng, Zhang Jian, Yang Rongxiang, and Liu Xiang. 2018. AR-HUD navigation system utilizing GPS data and navigation method.

[4] Kyong-Ho Kim Yoonsook Hwang, Byoung-Jun Park. 2016. Effects of Augmented-Reality Head-up Display System Use on Risk Perception and Psychological Changes of Drivers. *ETRI Journal, Vol 38, 4* (2016).