Truck Platooning Report

Shihab Ud Doula Email: shihab-ud.doula@stud.hshl.de
Lippstadt, Germany

Jasmeet Singh Matta Email: jasmeet-singh.matta@stud.hshl.de
Paderborn, Germany

Moshiur Rahman Prince Email: moshiour-rahman.prince@stud.hshl.de
Lippstadt, Germany

Contents

Abstract: This paper documents the development and implementation of a track platooning system. It covers the detection and handling of various scenarios, including coupling-decoupling, lane changing and steering, and single driver and system models with obstacles. The methodologies involve UML diagrams, UPPAAL models, and Python implementations.

1 Motivation

The track platooning project focuses on developing and implementing automated systems for managing and optimizing the behaviour of platoons of vehicles on tracks. This document outlines the project's objectives, methodologies, and outcomes, detailing the detection and handling of various scenarios such as coupling-decoupling, lane changing and steering, single driver and system model with obstacles.This project aims to enhance the safety, efficiency, and reliability of track-based transportation systems. By automating the detection and management of critical scenarios, we aim to reduce the risk of accidents, improve traffic flow, and minimize the need for manual intervention. This project leverages advanced modelling techniques and real-time data processing to achieve these goals.[Zh24]

2 LITERATURE REVIEW

Track platooning has been a research subject in the context of road and rail transportation. Previous studies have explored various aspects of platooning, including communication protocols, control algorithms, and safety mechanisms. The literature reveals the potential of using UML and UPPAAL models to formalize system behaviour and ensure robust performance. This project builds on these foundations by integrating scenario-specific models and implementing them in Python for real-world applicability.

3 PROBLEM STATEMENT[TJS16, Al15]

The project addresses several critical scenarios vital for the successful implementation of track platooning:

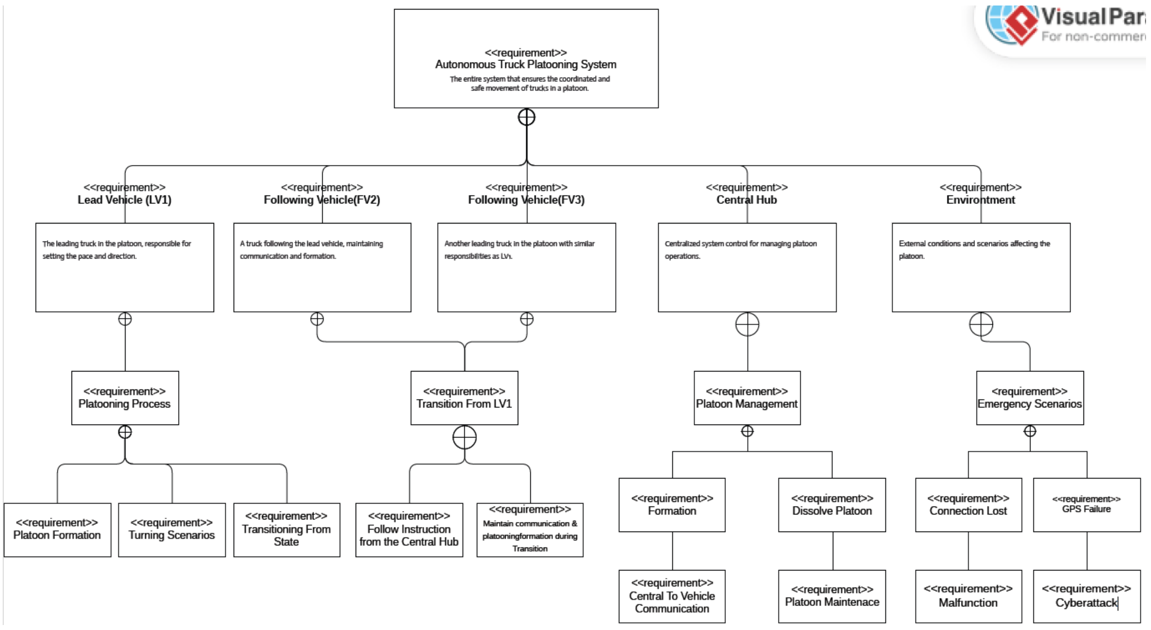
- **Obstacle Avoidance**: To detect obstacles and apply brakes accordingly to stop the vehicle or slow it down completely.
- **Lane Changing and Steering**: Managing lane changes and steering actions to prevent collisions and ensure smooth transitions.
- **Environmental Model**: Simulating the behaviour of an overall system and how the system will react to the changes in the environment

These Implementations are shown in Task 1 - Task 5 below.

4 TASK 1(UML Diagram For Scenarios)

4.0.1 Requirement Diagram (Truck Platooning)

The requirements diagram outlines the components of the Autonomous Truck Platooning System. It includes leading and following vehicles, the central hub, and the environment. Each element has specific requirements, such as platoon formation, communication, management, and handling emergency scenarios like connection loss, GPS failure, malfunctions, and cyberattacks.



<<requirement>> Autonomous Truck Platooning System
The entire system that ensures the coordinated and safe movement of trucks in a platoon.
<<requirement>> Lead Vehicle (LV1)
The leading truck in the platoon, responsible for setting the pace and direction.
<<requirement>> Following Vehicle(FV2)
A truck following the lead vehicle, maintaining communication and formation.
<<requirement>> Following Vehicle(FV3)
Another leading truck in the platoon with similar responsibilities as LV1.
<<requirement>> Central Hub
Centralized system control for managing platoon operations.
<<requirement>> Environtment
External conditions and scenarios affecting the platoon.
<<requirement>> Platooning Process
<<requirement>> Transition From LV1
<<requirement>> Platoon Management
<requirement>> Emergency Scenarios
<<requirement>> Platoon Formation
<<requirement>> Turning Scenarios
<<requirement>> Transitioning From State
<<requirement>> Follow Instruction from the Central Hub
<<requirement>> Maintain communication & platooningformation during Transition
<<requirement>> Formation
<<requirement>> Dissolve Platoon
<<requirement>> Connection Lost
<<requirement>> GPS Failure
<<requirement>> Central To Vehicle Communication
<<requirement>> Platoon Maintenace
<<requirement>> Malfunction
<<requirement>> Cyberattack
Visual Par
For non-commer

Fig. 1: Requirements Diagram For The System

4.0.2 Scenario 1(Obstacle Avoidance)

This sequence diagram in Figure 2 shows how a truck manages system malfunctions. It runs system diagnostics, acknowledges the malfunction, and exits to the nearest rest point. This ensures the car can safely handle unexpected obstacles or system issues.

4.0.3 Scenario 2(Lane Changing And Steering)

This sequence diagram in Figure 3 shows a driver requesting lane changes and the system's response. It involves checking lane availability, executing the lane change, and returning to

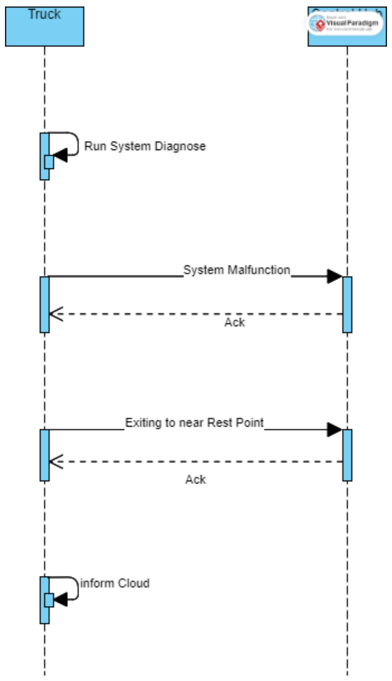


Fig. 2: System Diagnose Sequence Diagram

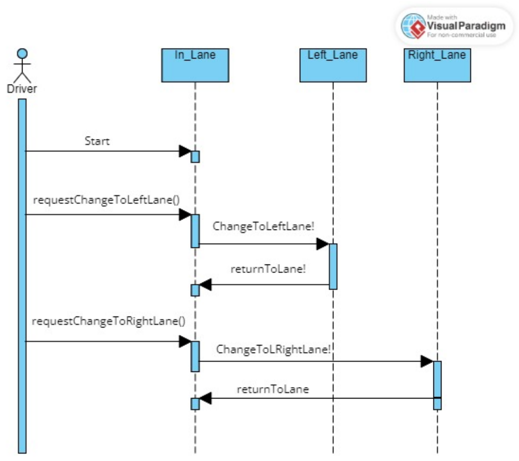


Fig. 3: Lane changing and steering Sequence diagram

the original lane, highlighting the system's ability to dynamically manage steering and lane changes.

4.0.4 Scenario 3(Environmental Model)

The diagrams in Figure 4 and Figure 5 depict how trucks manage lost connections and emergencies. They show reconnecting, informing the cloud, and handling emergencies by sending instructions and coordinating responses, ensuring the system's resilience and safety in dynamic environments.

5 TASK 2 and 3 (UPPAAL model for scenarios)

The whole model still needs to be implemented together as a whole without deadlock.

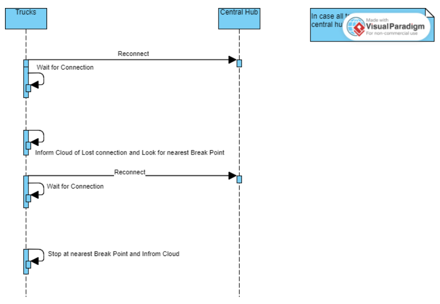

Trucks
Central Hub
In case all t
central hu
Reconnect
Wait for Connection
Inform Cloud of Lost connection and Look for nearest Break Point
Reconnect
Wait for Connection
Stop at nearest Break Point and Infrom Cloud

Fig. 4: Lost connection Sequence diagram

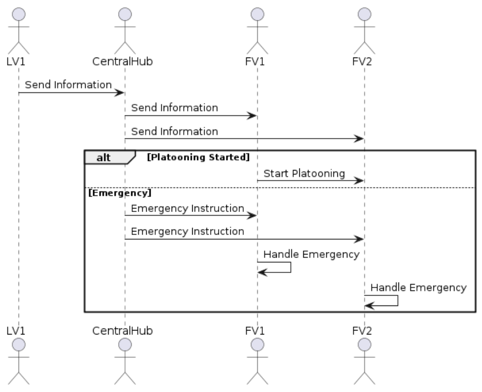


Fig. 5: Emergency Sequence diagram

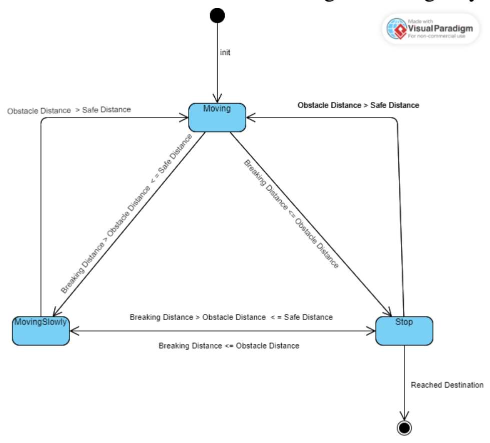

init
Obstacle Distance > Safe Distance
Obstacle Distance > Safe Distance
Moving
Breaking Distance > Obstacle Distance < = Safe Distance
Breaking Distance <= Obstacle Distance
MovingSlowly
Stop
Breaking Distance > Obstacle Distance < = Safe Distance
Breaking Distance <= Obstacle Distance
Reached Destination

Fig. 6: Obstacle avoidance State Machine Diagram

5.1 Scenario 1(Obstacle Avoidance)

This model(figure 7,8) shows a truck detecting and responding to obstacles. The truck moves, detects obstacles, adjusts its speed, and decides to stop or slow down based on the obstacle's distance. This ensures safe navigation through dynamic environments. It also shows the leader truck and the following trucks managing obstacles. The leader truck detects obstacles and adjusts speed, while the following trucks respond accordingly. This coordination maintains safety and efficiency in the platoon, preventing collisions and managing traffic flow.

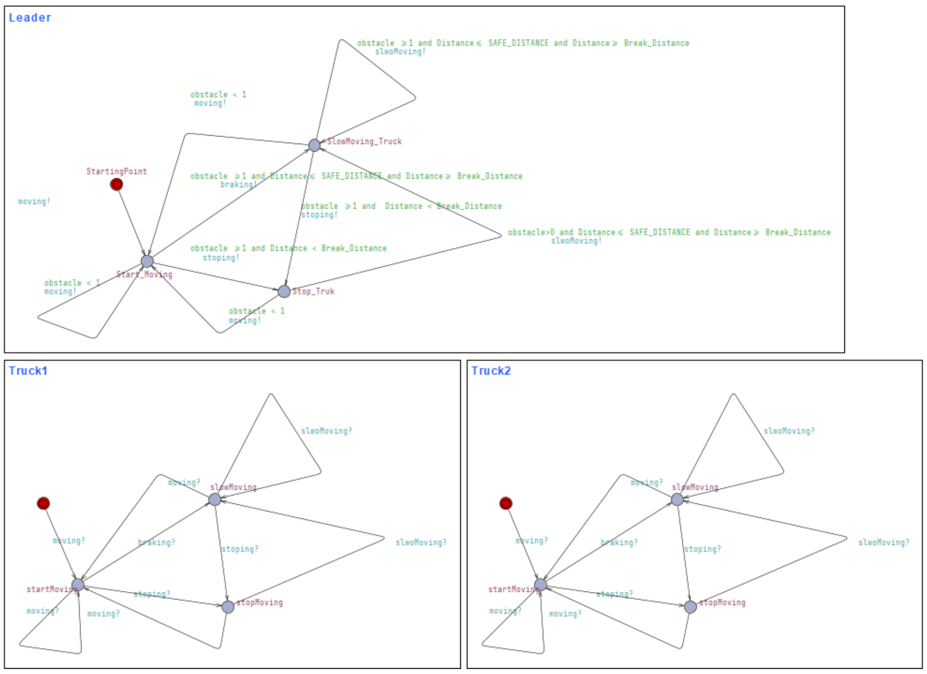


Fig. 7: Obstacle Uppaal 1

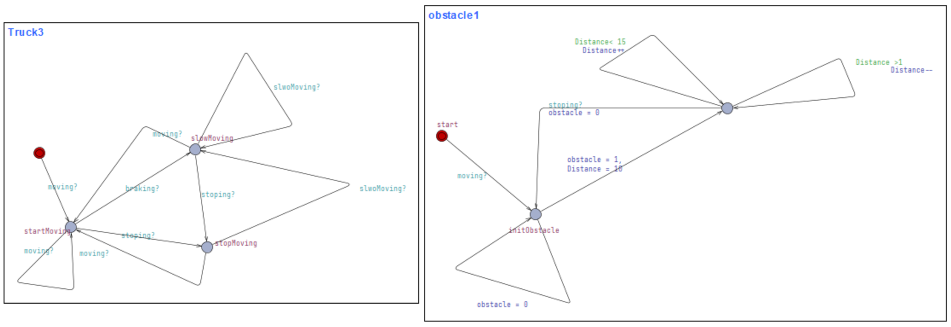


Fig. 8: Obstacle Uppaal 2

5.2 Scenario 2(Lane Changing Scenario)

This model, shown in Figure 9, illustrates how a vehicle changes lanes. It includes checking lane availability, executing lane changes, and ensuring safe transitions back to the original lane. The system handles standard steering and lane changes, allowing the vehicle to navigate effectively and safely.

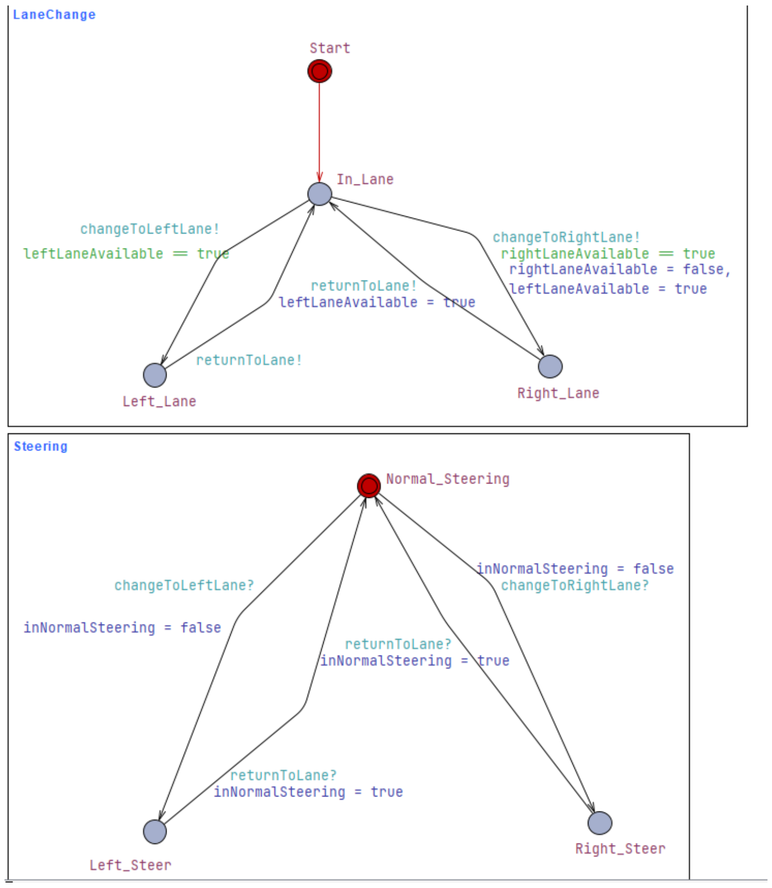


Fig. 9: Lane Change and Steering Uppaal

5.3 Scenario 2(Lane Changing Scenario)

This model depicts the overall system, showing how components interact within various environmental conditions. It includes lane changing, obstacle avoidance, and platoon formation. The model ensures vehicles can dynamically respond to environmental changes and maintain safe operations.

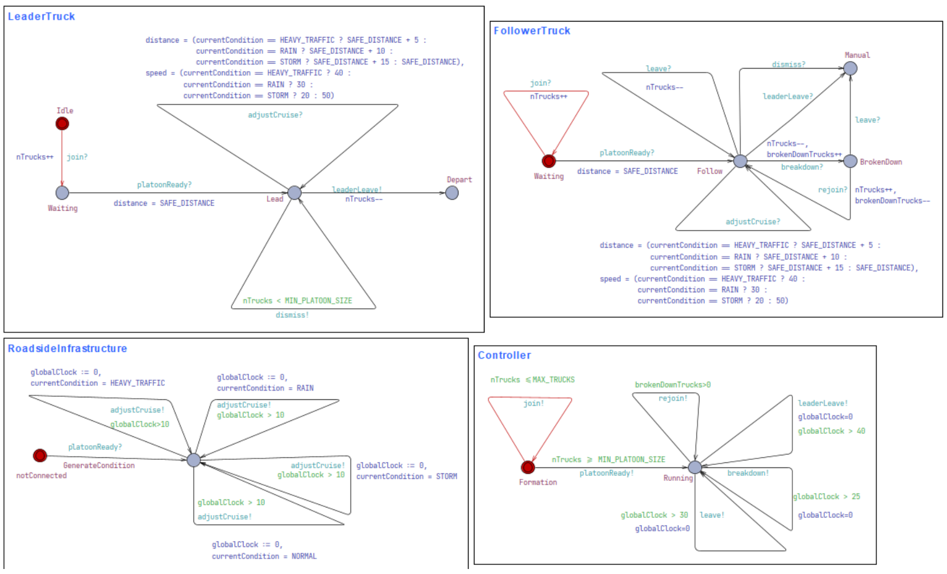


Fig. 10: System Model Uppaal

6 TASK 4

6.1 Decision Tree for Determining the Leader Vehicle[Su16]

To determine the leader vehicle in our truck platooning system, we implemented a decision tree classifier using Scikit-learn in Google. Below is the decision tree generated based on our dataset in Figure 11.

Decision Tree Visualization:

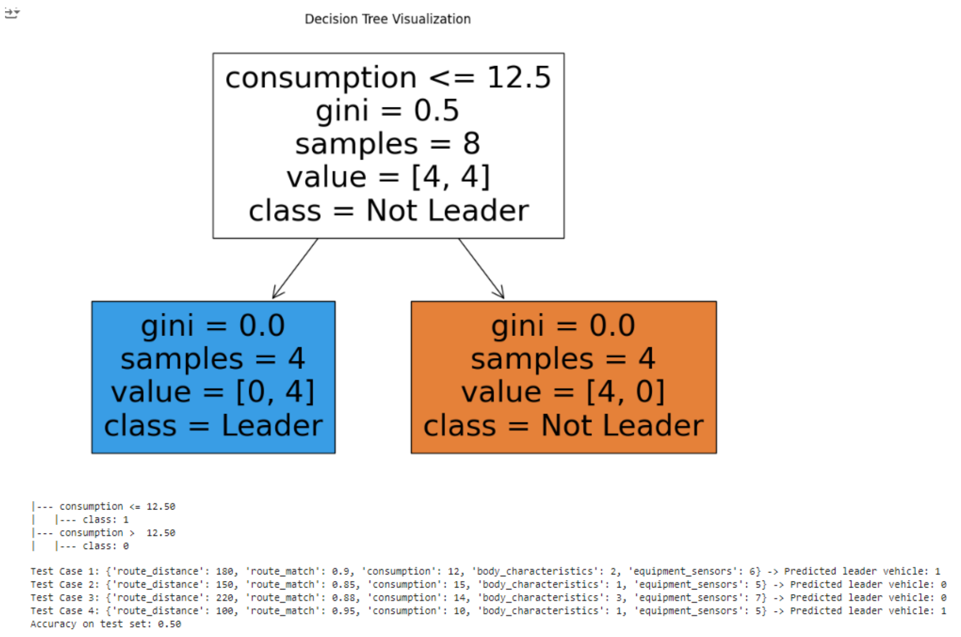
Key Code Snippet:

Whole code is in GITHUB Explanation:

Feature Selection: We selected relevant features such as route distance, route match, consumption, body characteristics, and equipment sensors.

Data Splitting: The dataset was split into training and testing sets to evaluate the model's performance.

Model Training: A DecisionTreeClassifier was initialized with a maximum depth of 3 to ensure the tree remains interpretable and trained on the training set.



```
|--- consumption <= 12.50
|   |--- class: 1
|--- consumption >  12.50
|   |--- class: 0

Test Case 1: {'route_distance': 180, 'route_match': 0.9, 'consumption': 12, 'body_characteristics': 2, 'equipment_sensors': 6} -> Predicted leader vehicle: 1
Test Case 2: {'route_distance': 150, 'route_match': 0.85, 'consumption': 15, 'body_characteristics': 1, 'equipment_sensors': 5} -> Predicted leader vehicle: 0
Test Case 3: {'route_distance': 220, 'route_match': 0.88, 'consumption': 14, 'body_characteristics': 3, 'equipment_sensors': 7} -> Predicted leader vehicle: 0
Test Case 4: {'route_distance': 100, 'route_match': 0.95, 'consumption': 10, 'body_characteristics': 1, 'equipment_sensors': 5} -> Predicted leader vehicle: 1
Accuracy on test set: 0.50
```

Fig. 11: Decision Tree for Leader Vehicle Selection

```python
# Define features and target variable
X = df[['route_distance', 'route_match', 'consumption', 'body_characteristics', 'equipment_sensors']]
y = df['leader_vehicle']
```

Fig. 12: Code

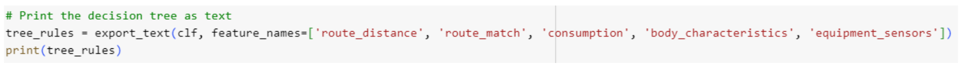
```python
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the Decision Tree Classifier with max depth to prune the tree
clf = DecisionTreeClassifier(random_state=42, max_depth=3)

# Train the model
clf.fit(X_train, y_train)
```

Fig. 13: Code

```python
# Visualize the Decision Tree
plt.figure(figsize=(12, 8))
plot_tree(clf, feature_names=['route_distance', 'route_match', 'consumption', 'body_characteristics', 'equipment_sensors'], class_names=['Not Leader', 'Leader'], filled=True)
plt.title('Decision Tree Visualization')
plt.show()
```

Fig. 14: Code

Tree Visualization: The decision tree was visualized to understand the criteria for predicting the leader vehicle.



```python
# Print the decision tree as text
tree_rules = export_text(clf, feature_names=['route_distance', 'route_match', 'consumption', 'body_characteristics', 'equipment_sensors'])
print(tree_rules)
```

Fig. 15: Code

This decision tree helps us understand the criteria for selecting a leader vehicle in our truck platooning system. By analyzing the tree, we can ensure that the decisions align with our system requirements and optimize the platoon's overall efficiency and safety.

7 TASK 5

8 Conclusion

9 Declaration of Originality

We, herewith, declare that we have composed the present paper and work by ourselves and without using anything other than the cited sources and aids. Sentences or parts of sentences quoted are marked as such; full details of the publications concerned indicate other references about the statement and scope. The paper and work in the same or similar form have not been submitted to any examination body or published. This paper has not yet, even in part, been used in another examination or as a course performance. We agree that a plagiarism checker may check my work.

Date&Place - Shihab Ud Doula, Jasmeet Singh Matta, Moshiur Rahman Prince

Bibliography

[Al15] Alam, Assad; Besselink, Bart; Turri, Valerio; Mårtensson, Jonas; Johansson, Karl H: Heavy-duty vehicle platooning for sustainable freight transportation: A cooperative method to enhance safety and efficiency. IEEE Control Systems Magazine, 35(6):34–56, 2015.

[Su16] Suthaharan, Shan: Machine learning models and algorithms for big data classification. Integr. Ser. Inf. Syst, 36:1–12, 2016.

[TJS16] Tsugawa, Sadayuki; Jeschke, Sabina; Shladover, Steven E: A review of truck platooning projects for energy savings. IEEE Transactions on Intelligent Vehicles, 1(1):68–77, 2016.

[Zh24] Zhang, Tianya Terry; Jin, Peter J; McQuade, Sean T; Bayen, Alexandre; Piccoli, Benedetto: Car-following models: A multidisciplinary review. IEEE Transactions on Intelligent Vehicles, 2024.