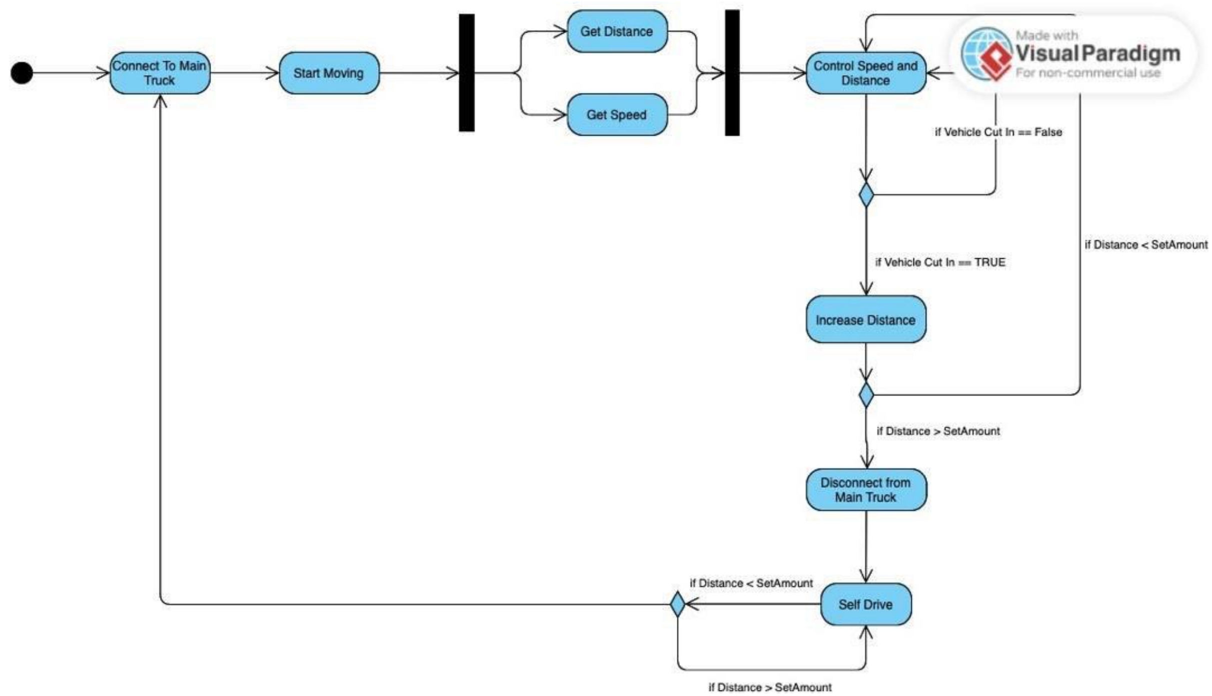
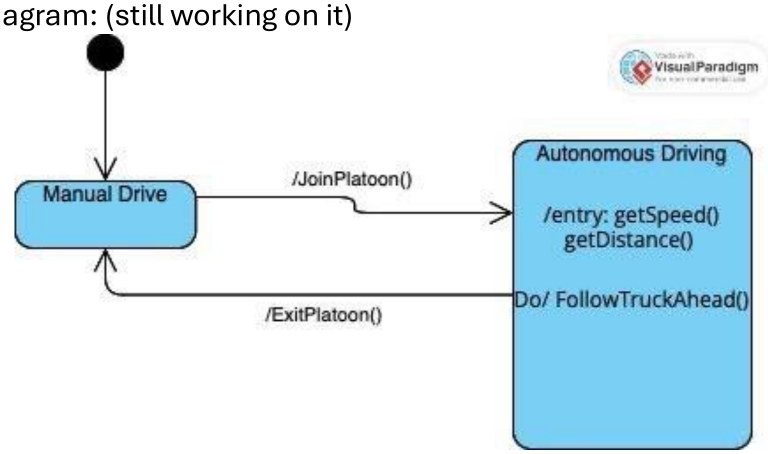
DPS PROJECT
GROUP ONE

First let's look at the diagrams that we prepared initially:

Activity Diagram:



State machine diagram: (still working on it)



Task 1: Identify Data/Signals/Events and Specify a Protocol

Step 1: Identify Required Data/Signals/Events
For the trucks in a platoon to interact and communicate effectively, the following types of data and events are necessary:

1. **Vehicle Data (from each truck):**
   - **Speed**: Current speed of the truck.
   - **Acceleration/Deceleration**: Rate of speed change.
   - **Position**: GPS coordinates or relative position in the platoon.
   - **Direction**: Current direction of travel.
   - **Fuel/Battery Level**: Status to ensure the truck can continue in the platoon.
2. **Control Signals:**
   - **Distance to Preceding Truck**: To maintain a safe gap.
   - **Braking Commands**: Emergency or routine.
   - **Lane Change Requests**: For overtaking or exiting the platoon.
   - **Join/Leave Platoon Request**: Signaling intent to join or leave.
3. **Events for Communication:**
   - **Heartbeat Signals**: Periodic messages to ensure active communication.
   - **Failure Notifications**: Alerts in case of sensor or communication issues.
   - **Synchronization Signals**: For starting or ending specific maneuvers.

Step 2: Specify an Appropriate Protocol
The communication protocol must be:

- **Robust**: Handle failures gracefully.
- **Real-time**: Ensure low latency for critical commands.
- **Efficient**: Minimize bandwidth use.

Suggested Protocol:

1. **Protocol Layers**:
   - **Application Layer**: Defines the types of messages (e.g., speed update, join request).
   - **Transport Layer**: Use UDP for real-time communication with a reliability mechanism added for critical data.
   - **Network Layer**: IP-based routing for inter-truck communication.
2. **Message Types**:
   - **Data Update Messages**: Periodically transmit speed, acceleration, and position.
   - **Control Commands**: Transmit braking, lane change, or joining requests.
   - **Acknowledgments**: Confirm receipt of critical commands.
3. **Structure of a Message**: Each message could have fields such as:
   - **Message Type**: Identifies the purpose (e.g., SPEED_UPDATE, EMERGENCY_BRAKE).
   - **Sender ID**: Unique identifier for the truck sending the message.
   - **Target ID**: Specifies the intended recipient(s).
   - **Timestamp**: Ensures events are processed in order.

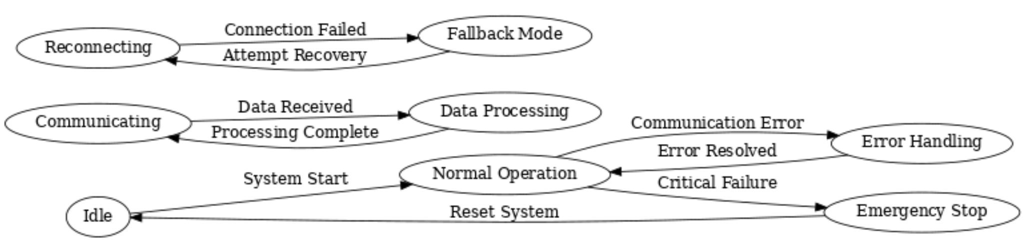
- **Payload**: Contains the actual data (e.g., speed value).
- **Checksum**: Validates the integrity of the message.

4. **Communication Framework**:
    - **Broadcast for Regular Updates**:
        - Trucks broadcast their speed, position, and distance to neighbors.
    - **Unicast for Specific Commands**:
        - E.g., Lane change requests sent directly to the preceding and following trucks.
    - **Failure Recovery Mechanism**:
        - On heartbeat timeout, initiate a fail-safe mode (e.g., increase distance or slow down).

Step 3: Model Using State Machines
Use **State Machines** to represent the behavior of the system. A high-level representation could include:

1. **States**:
    - **Normal Operation**: Trucks communicate and maintain safe distances.
    - **Adjusting Distance**: Adapting to speed changes of the leading truck.
    - **Error Recovery**: Handling communication failure or system faults.
    - **Joining Platoon**: A new truck joins the platoon.
    - **Exiting Platoon**: A truck exits safely.
2. **Transitions**:
    - **Event: SPEED_UPDATE** -> Transition to Adjusting Distance.
    - **Event: FAILURE_NOTIFICATION** -> Transition to Error Recovery.
    - **Event: JOIN_REQUEST** -> Transition to Joining Platoon.
    - **Event: EXIT_REQUEST** -> Transition to Exiting Platoon.

UML DESIGN FOR TASK 1:



Task 2: Identify Control Behavior for Trucks
This task focuses on ensuring that the trucks in the platoon maintain safe distances and handle failures gracefully. Below is a detailed, simplified explanation:

Step 1: Control Behavior for Maintaining Distance
Objective: Ensure the distance between trucks is maintained dynamically, based on speed and real-time data.
How it works:

- **Sensors and Data**: Each truck is equipped with sensors (e.g., LiDAR, radar) to measure the distance to the truck ahead.
- **Dynamic Adjustment**: Trucks adjust their speed using acceleration or braking to maintain a safe following distance.
- **Formula for Safe Distance**:
  Safe Distance=Reaction Time×Speed+Braking DistanceSafe Distance=Reaction Time×Speed+Braking Distance
    - Reaction Time: Delay for processing signals.
    - Braking Distance: Depends on speed and road conditions.

Implementation Logic:

1. **Data Collection**:
    - Measure the distance to the preceding truck.
    - Receive speed and acceleration data via communication protocols.
2. **Decision Making**:
    - If the distance is less than the safe threshold, apply braking or reduce speed.
    - If the distance is too large, accelerate to close the gap.
3. **Feedback Loop**:
    - Continuously monitor and adjust based on real-time data.

Step 2: Robustness Against Failures
Objective: Ensure the system remains stable during communication or sensor failures.
Types of Failures:

1. **Communication Failures**:
    - Missing data or delayed updates.
    - No heartbeat signals from the preceding truck.
2. **Sensor Failures**:
    - Faulty distance measurement.
3. **Control Failures**:
    - Trucks unable to execute braking or acceleration commands.

Strategies for Robustness:

1. **Fallback Mechanisms**:
    - In case of communication failure:
        - Increase distance automatically.
        - Switch to sensor-only mode.
    - If a sensor fails:
        - Use averaged data from preceding communications.

2. **Fail-Safe States**:
   - If critical failures occur, reduce speed to a minimum and alert the driver.
3. **Redundancy**:
   - Use multiple sensors and communication channels to ensure reliability.

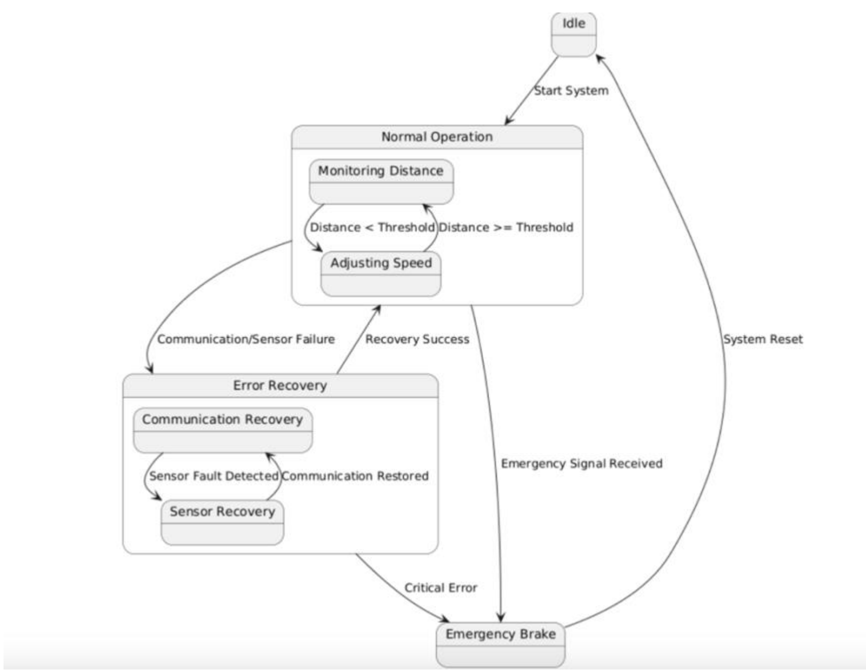
Step 3: Model-Based Specification Using State Machines
Use **UML State Machines** to describe control behavior.
States:

1. **Normal Operation**: Trucks communicate and maintain safe distances.
2. **Adjusting Distance**: Adapting to changes in the lead truck's speed.
3. **Error Recovery**: Handling communication or sensor failures.
4. **Emergency Brake**: Triggered by sudden stops or critical errors.
5. **Idle**: Waiting state when a truck is parked or disconnected.

Transitions:

- **Distance Warning**: Transition from Normal Operation to Adjusting Distance when safe distance is breached.
- **Communication Failure**: Transition to Error Recovery if heartbeat signals are lost.
- **Critical Error**: Transition to Emergency Brake if control systems fail.

Step 4: UML Diagram Code



Explanation of UML DIAGRAM

1. Nested States in Normal Operation

- **Monitoring Distance**: Continuously measures the distance from the preceding truck.
- **Adjusting Speed**: Engages when the distance breaches a threshold. This modular separation clarifies that monitoring and adjusting are tightly coupled but logically distinct activities.

2. Error Recovery Substates

- **Communication Recovery**: Dedicated state to attempt re-establishing communication, such as resending packets or switching to fallback channels.
- **Sensor Recovery**: Handles sensor faults, potentially using redundant sensors or estimating data from historical patterns.

This breakdown ensures that error recovery processes are modular and easier to debug.

3. Transitions Between Substates

- Specific conditions for moving between states (e.g., resolving a sensor failure after a communication failure) make the system robust and better aligned with real-world scenarios.

4. Emergency Brake as an Isolated State

- The emergency brake is separated to emphasize that it's a critical state triggered by high-priority events like:
    - Sudden stops by the lead truck.
    - Multiple subsystem failures (e.g., communication and sensor).
- The system resets only after resolving the cause.

5. Idle State for Initialization and Reset

- Idle acts as the default state for the system, allowing:
    - Initialization of subsystems before entering normal operation.
    - Safe reset after resolving emergencies.