Smart Assistant Drive

Andrea Brighi
Daniele Di Lillo

Goal

create a smart assistant drive involving Multi Agent Systems

- Digitized version of the speed signs to create a Intelligent Speed Assistance
- Digitized road with lanes and driving flows to create a virtual environment for cars
- Digitized traffic lights to regulate junctions and to be recognizable by cars

Requirements

- The simulation start from a predefined scenario
- The cars should start going from the starting point A to the final point B, in a shared environment with other cars and junctions
- Cars should not create crashes or violate the road laws
- Every single car must keep a security distance from the next one
- Every junction is regulated by a semaphore, including 4 lights (one for each direction), and must be temporized

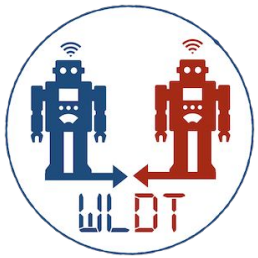
Implementation requirements

- Track the cars' information into a specific traffic Digital Twin, to calculate the distances and manage the traffic queues
- Track the semaphore states and events into a Digital Twin
- Use agents to model the behavior of cars and coordinate the junction's traffic lights

ASSUMPTIONS:

- GPS for cars and Internet connection for all elements
- The cars have radar for near collision
- Lane assist and a camera to recognize road lanes

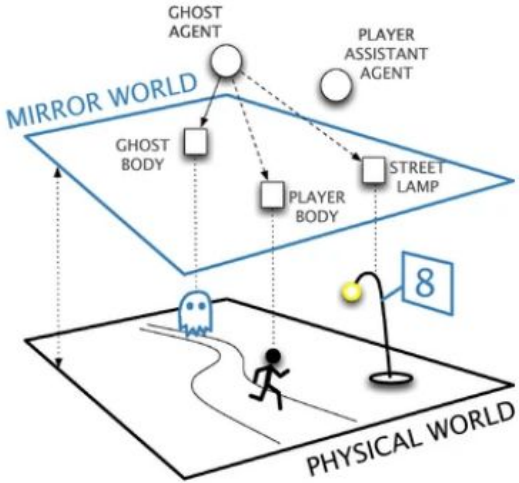
Technologies



DESIGN

System architecture

The overall architecture is inspired by the Agents & Artifacts model and by principles of Intelligent Pervasive Systems.



Digital Twins can be seen as artifacts that mediate the interaction between agents and the environment, providing observable properties and executable operations of the physical world.

Cyber Physical Multi Agent System

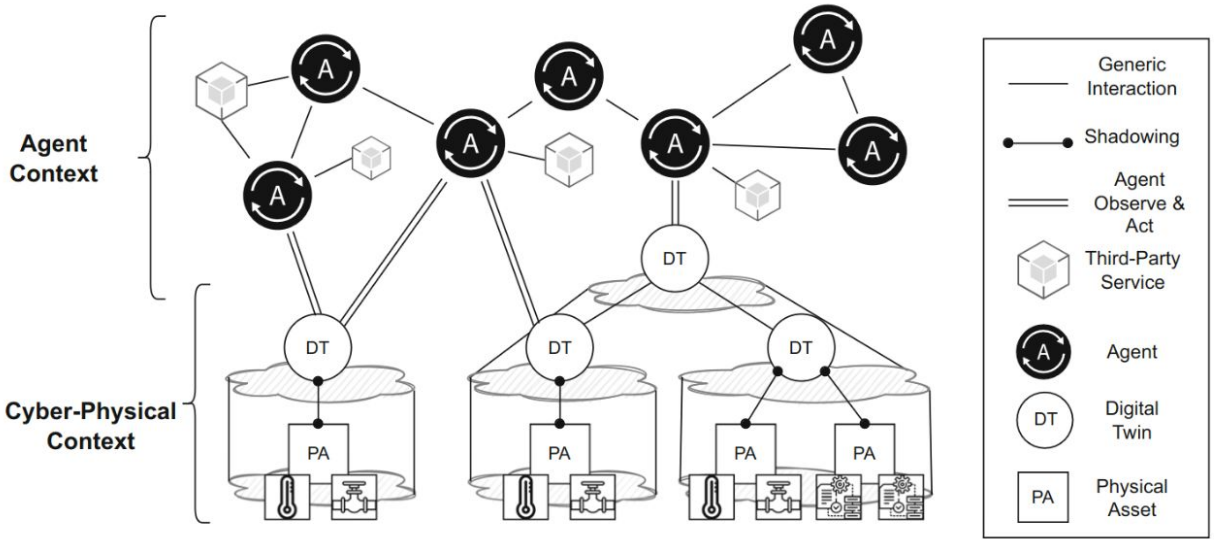


Fig. 1 *Separation of concerns*: DTs operate within the boundaries set by the *local context* of the associated PA, whereas agents operate within the boundaries of the *global context* of the whole application set by the application designer

Multi Agent Systems involved

Car MAS

It manages a car agent in the simulation, has limited vision, and perceives the environment using road and signs microservices and the Digital Twins to get updates about traffic lights and the car ahead (not known by the agent)

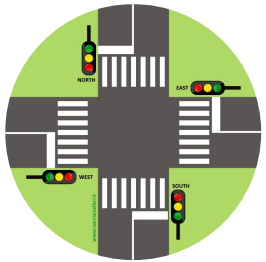
Junction MAS

Manages all traffic light agents in the simulation, grouped together in junctions with independent timing

A & A inspiration

Evaluate benefits and use cases for Multi-Agent Systems and Digital Twins.

Agents



Junction agent

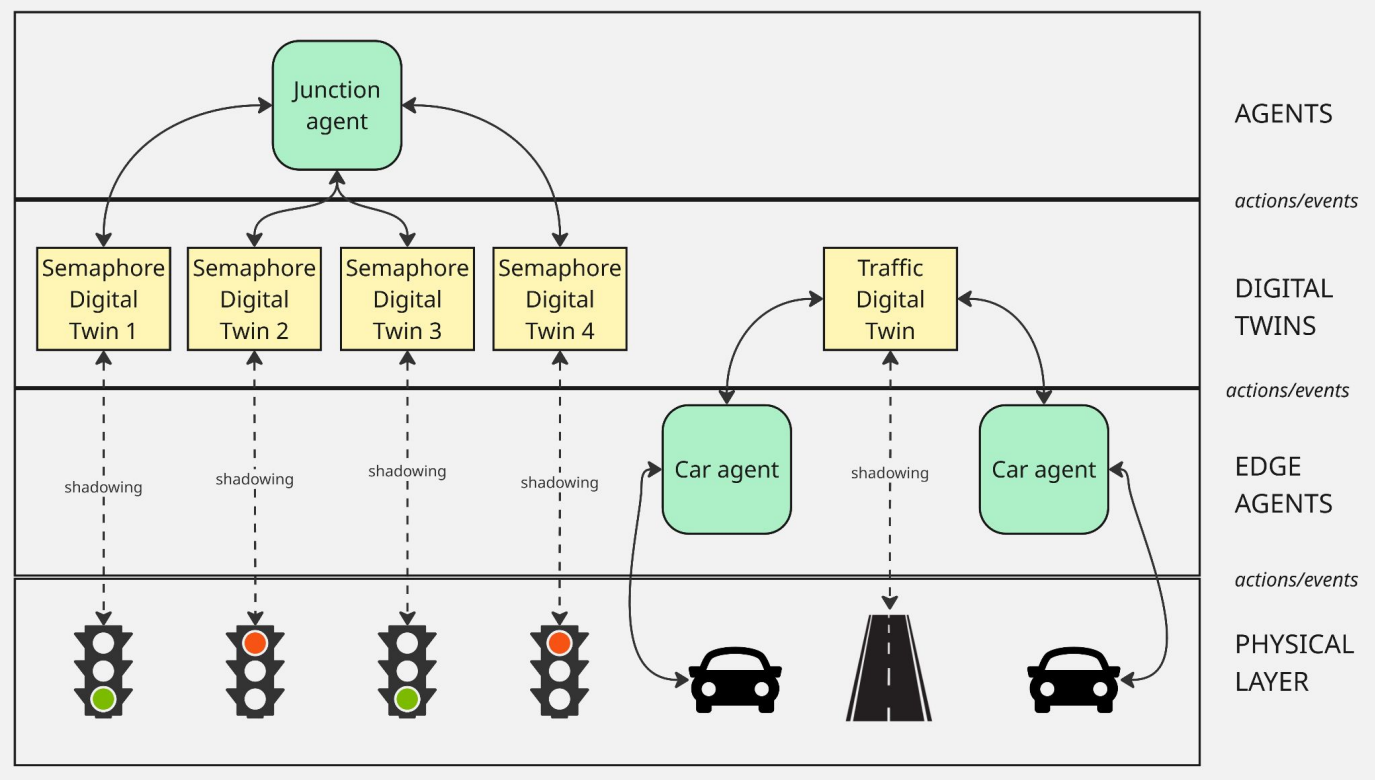
Car agent

Artifacts

Traffic light DT

Traffic DT

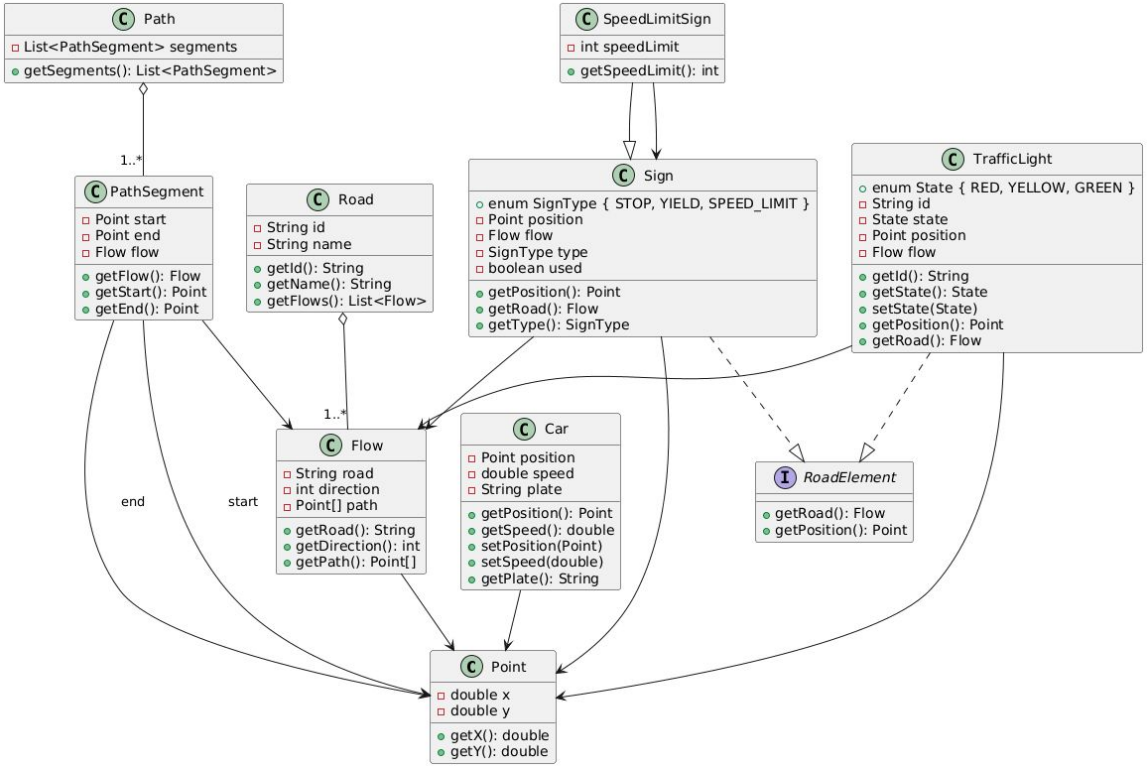
Implemented architecture



Agents to the edge

We also use the concept of edge agents that are agents directly running on the physical object (like IoT) in order to have continues and immediate perception and control on the environment (no network delay or problems).

Model


Path
- List<PathSegment> segments
+ getSegments(): List<PathSegment>
SpeedLimitSign
- int speedLimit
+ getSpeedLimit(): int
1..*
PathSegment
- Point start
- Point end
- Flow flow
+ getFlow(): Flow
+ getStart(): Point
+ getEnd(): Point
Road
- String id
- String name
+ getId(): String
+ getName(): String
+ getFlows(): List<Flow>
Sign
+ enum SignType { STOP, YIELD, SPEED_LIMIT }
- Point position
- Flow flow
- SignType type
- boolean used
+ getPosition(): Point
+ getRoad(): Flow
+ getType(): SignType
TrafficLight
+ enum State { RED, YELLOW, GREEN }
- String id
- State state
- Point position
- Flow flow
+ getId(): String
+ getState(): State
+ setState(State)
+ getPosition(): Point
+ getRoad(): Flow
1..*
Flow
- String road
- int direction
- Point[] path
+ getRoad(): String
+ getDirection(): int
+ getPath(): Point[]
Car
- Point position
- double speed
- String plate
+ getPosition(): Point
+ getSpeed(): double
+ setPosition(Point)
+ setSpeed(double)
+ getPlate(): String
RoadElement
+ getRoad(): Flow
+ getPosition(): Point
end
start
Point
- double x
- double y
+ getX(): double
+ getY(): double

Car MAS - 1

After retrieving all the starting information, the car agent has two goals:

- Reaching the destination
- Drive safely on the roads if it is in autonomous driving

Drive safety:

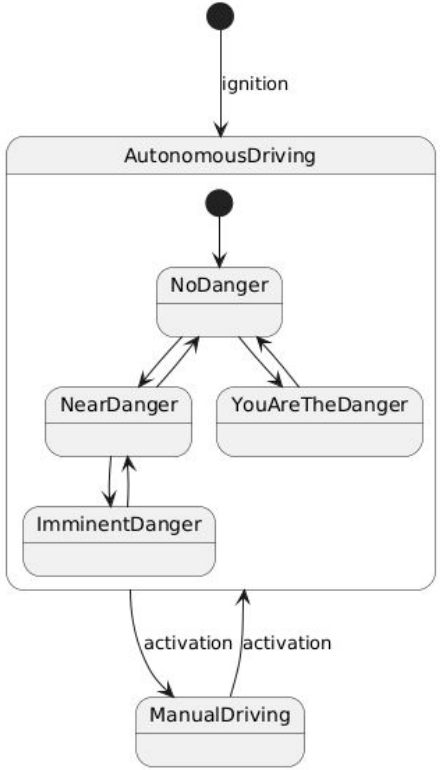
- Keep a speed below the speed limit (given by the last speed limit sign)
- Keep a safe distance (based on its speed) from the car ahead
- Stop at Stop signs and red traffic lights
- The car should prefer the gentle slowing down (without using the brakes) to using the brakes

Car MAS - 2

This gives the agent 4 states:

- Immediate danger
- Near danger
- No danger
- You are the danger

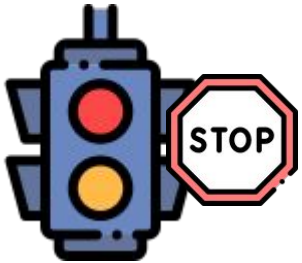
The ”You are the danger” state is applied only when the agent is too slow, so it must speed up.



Car MAS - Scenarios

Road scenarios

1) Car exceeding the speed limit
2) Car get closer to a traffic light (red or yellow) or stop sign
3) Near car ahead



Everything is based on distance and speed and time.

Need to translate driving rule in math concept and limit.

- Optimal safe distance (based on car speed and other car speed)
- Minimum safe distance (Braking distance)

Car MAS - Rules

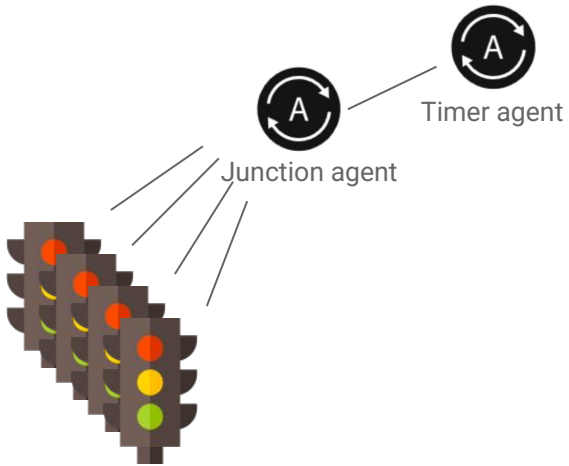
```
distance_to_stop(CS, DECEL, SD) :-
        reaction_time(RT)
        & SD = (CS * CS) / (2 * DECEL) + (CS * RT).

stopping_distance(CS, SD) :- break_speed(BS) &
        distance_to_stop(CS, BS, SD).

dragging_distance(CS, SD) :- drag_speed(DS) &
        distance_to_stop(CS, DS, SD).

stopping(X1, Y1, X2, Y2, CS) :- distance(X1, Y1, X2, Y2, D)
                & stopping_distance(CS, SD)
                & D < SD + 10.

dragging(X1, Y1, X2, Y2, CS) :- distance(X1, Y1, X2, Y2, D)
                & dragging_distance(CS, SD)
                & D < SD + 10.
```

```
normal_safe_distance(S, SD) :- car_distance(D) &
        SD = S / 2 + 10 + D.

dynamic_safe_distance(CS, OS, SD) :-
        car_distance(D) &
        Diff = CS - OS &
        Diff > 0 &
        SD = D + 10 + Diff * 2.

dynamic_safe_distance(CS, OS, SD) :-
        Diff = CS - OS &
        Diff <= 0 &
        normal_safe_distance(S, SD).
```

Junction MAS - 1

This sub-system is engineered to handle the complexity of urban crossings through a coordinated effort between two different types of specialized agents:

- Junction Agent
- Timer Agent



Junction MAS - 2

Junction Agent

This agent acts as the local orchestrator for a specific crossing.

Manages a cluster of four traffic lights that are modeled as Digital Twins

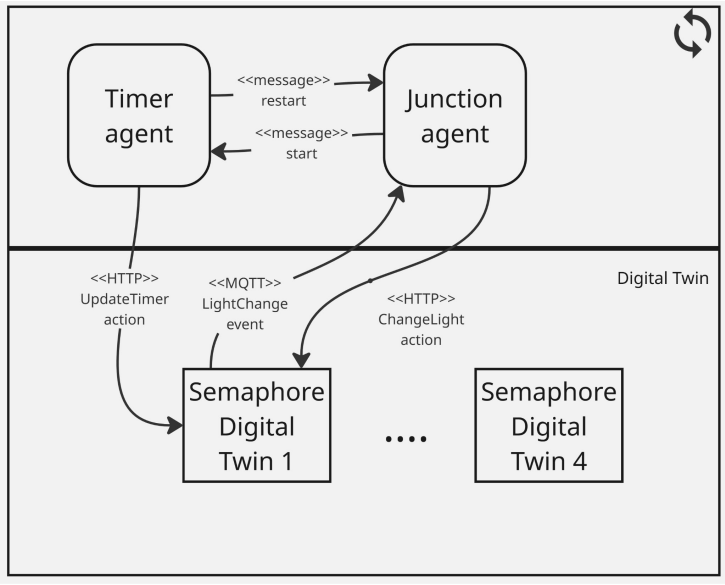
Timer Agent

Each intersection is assigned a dedicated Timer Agent.

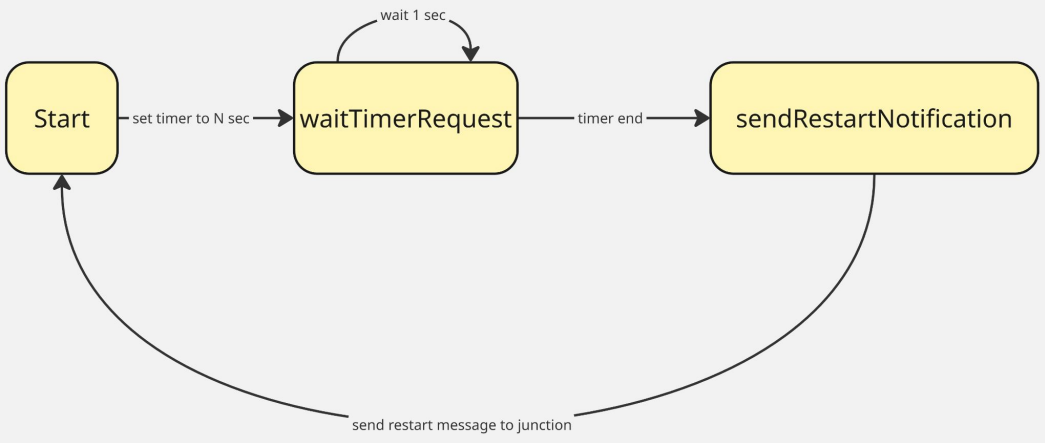
This agent is responsible for the temporal logic and to allow execution of different specific traffic policies

Junction MAS - 3

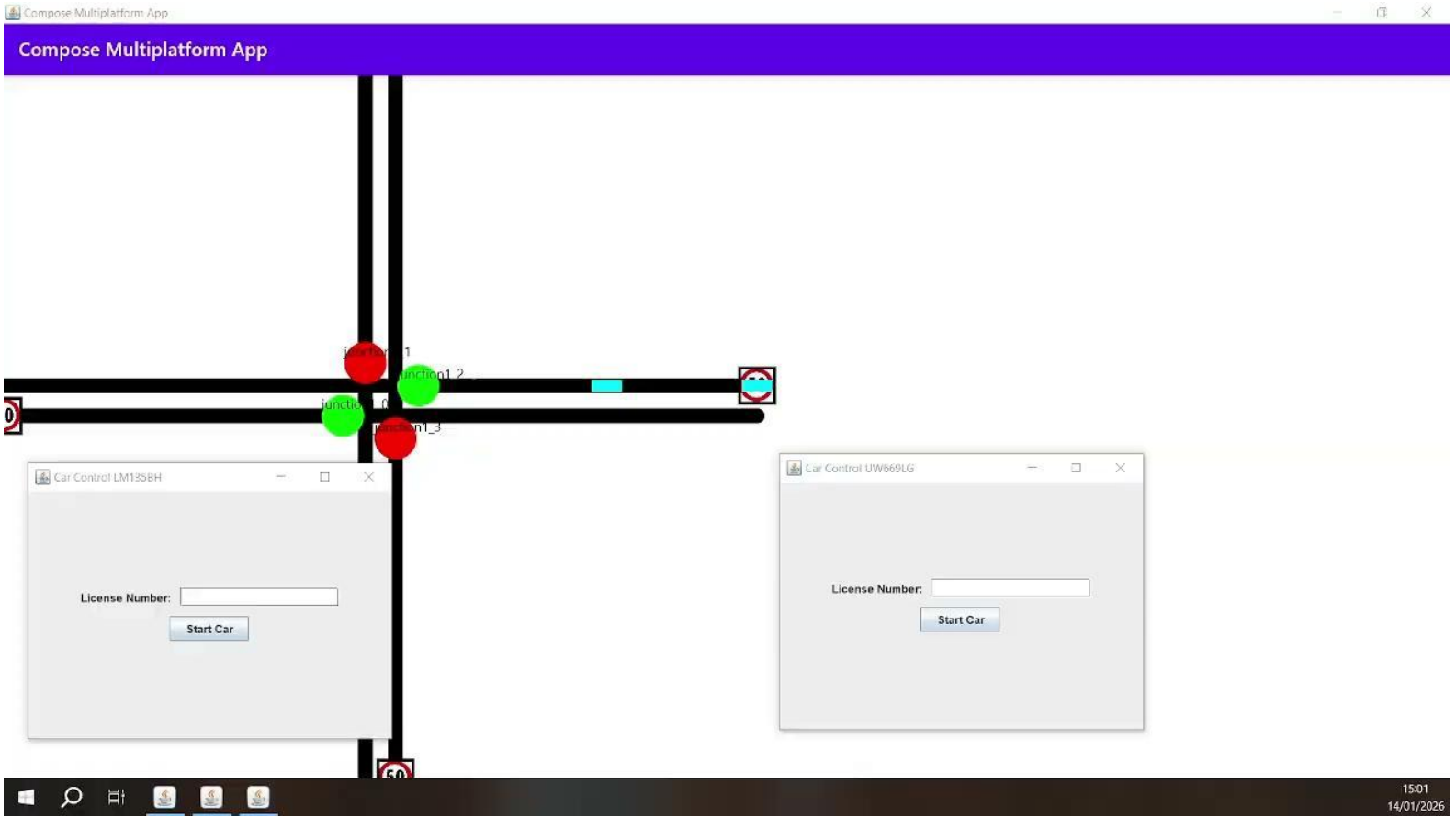
Junction MAS - 4



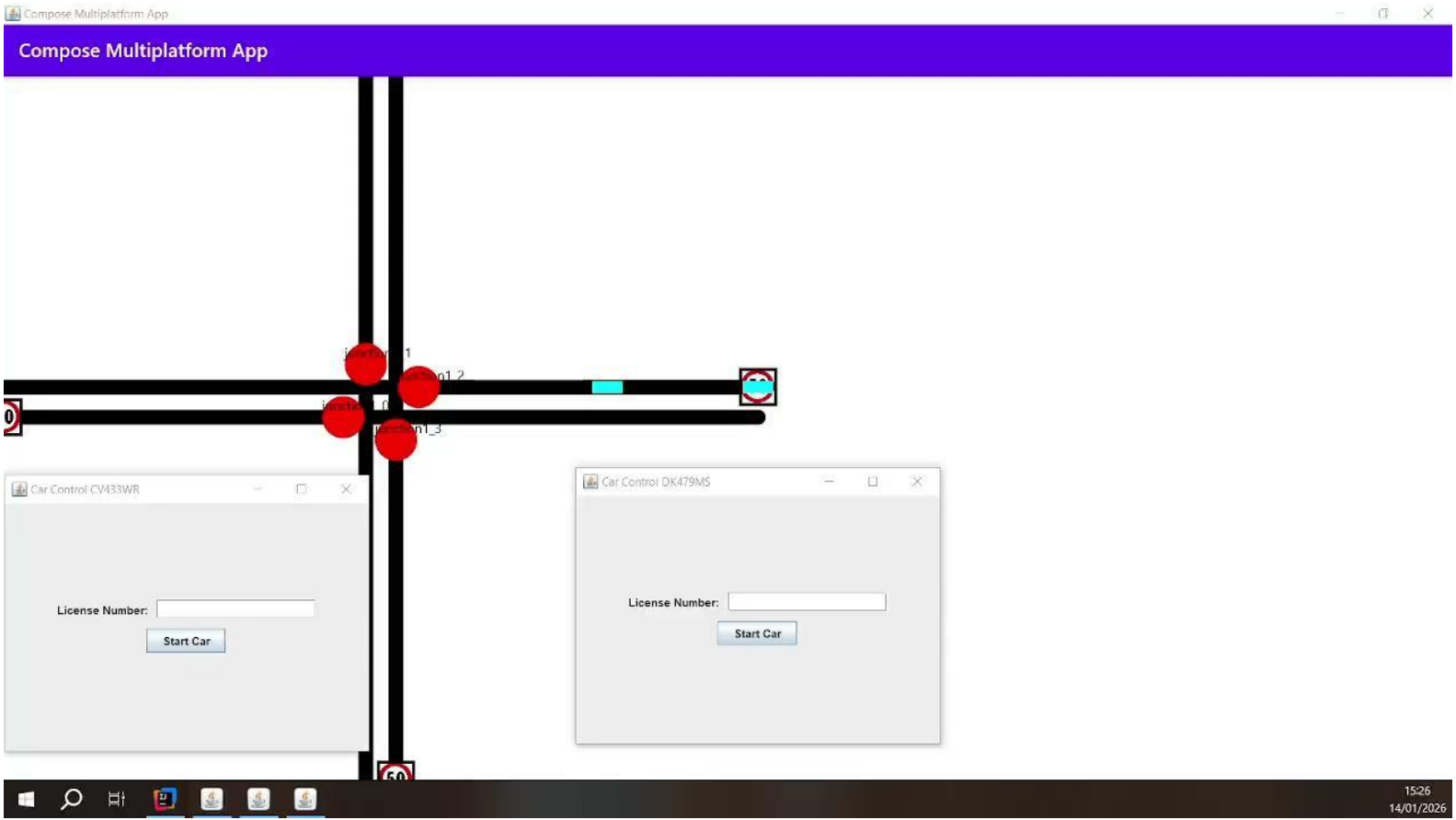
Agents and DTs



Timer agent states


Compose Multiplatform App
Compose Multiplatform App
Car Control LM135BH
License Number:
Start Car
Car Control UW869LG
License Number:
Start Car
15:01
14/01/2026

Video demo 1


Compose Multiplatform App
Compose Multiplatform App
Car Control CV433WR
License Number:
Start Car
Car Control DK479MS
License Number:
Start Car
50
15:26
14/01/2026

Video demo 2