Pitlane Command Protocol for Vehicular Networks on the Imola Circuit using SUMO, OMNeT++ and Veins

Marc Guitart Frescó
June 2025

Executive Summary

This project presents the design and implementation of a vehicular communication protocol simulating the "Box Box" command used in Formula 1 racing, where a roadside unit (RSU) instructs vehicles to enter the Pitlane. The simulation was carried out using SUMO, OMNeT++, and the Veins framework over a digital model of the Imola circuit. Five vehicles were deployed on the circuit and programmed to respond to RSU commands by altering their route dynamically toward the Pitlane.

The aim was to replicate realistic behavior where vehicles reduce speed to the regulated Pitlane limit and avoid collisions during the maneuver. Finite State Machines (FSMs) were designed for both the RSU and the autonomous vehicles to handle message transmission and behavioral transitions. Although the vehicles currently do not rejoin the main circuit after entering the Pitlane due to technical constraints, the protocol successfully demonstrates RSU-initiated route switching and speed adjustment under simulation.

The results validate that command reception and Pitlane entry operate as intended. Future improvements would include full Pitlane exit integration, handling of dynamic traffic, and metrics collection for performance benchmarking.

Index

1 Project Description

Project Description

This project focuses on simulating and managing Vehicle-to-Infrastructure (V2I) communications within a controlled Formula 1-like scenario, leveraging the Veins framework integrated with SUMO (Simulation of Urban MObility) and OMNeT++. Our primary goal has been to implement and validate a command-based routing protocol in which a roadside unit (RSU) instructs autonomous vehicles to deviate from the main racing trajectory and enter the Pitlane, mimicking the well-known “Box Box” instruction used in motorsports.

The first step was modeling the Imola circuit using SUMO’s internal mapping tool, which converts satellite-based visual layouts (e.g., from Google Maps) into usable *.net.xml* formats. The environment includes clearly defined segments for both the racing path and the Pitlane section, allowing differentiated routing and behavior.

Two main roles were defined in this architecture:

- **RSU (Road Side Unit)**: Acts as the control node that emits the “Box Box” message using a broadcast mechanism. It is responsible for initiating the route change command.
- **Vehicle**: Autonomous cars driving on the circuit. Upon receiving the RSU command, they switch from the main trajectory to a predefined Pitlane route and adapt their speed accordingly to comply with Pitlane regulations (80 km/h limit).

With the roles and interaction logic settled, Finite State Machines (FSMs) were designed for both the RSU and the vehicles. Each FSM includes a structured state-transition logic triggered by message reception or simulation events. The vehicle FSM handles transitions from normal racing behavior to Pitlane entry and deceleration, while the RSU FSM controls the emission of the command based on simulation time. The system currently does not implement Pitlane re-entry, but this limitation is noted for future work.

1.1 Project Objectives

The objective of this project is to develop and evaluate a command-based route-switching protocol in a simulated V2I environment using SUMO and Veins, specifically designed for a racing scenario. The project aims to:

- Design a robust RSU-triggered route-changing protocol using SUMO and OMNeT++.
- Define a message flow enabling autonomous vehicles to receive and act on external instructions dynamically.
- Implement FSMs to manage both infrastructure and vehicle behavior in response to timed or triggered events.
- Evaluate the system's correctness in terms of command reception, route compliance, and controlled speed transition into the Pitlane.
- Explore future improvements such as rejoining logic, integration of dynamic traffic, or probabilistic command reception scenarios.

The implementation is grounded on core configuration and logic files such as *omnetpp.ini, imola.rou.xml*, *rerouter.add.xml*, and *osm.net.xml.gz*, which define the simulation parameters, routing behavior, and road network respectively.

2 Protocol Design

2.1 System Architecture Overview

The system is composed of three main integrated components: the mobility simulator SUMO, the network simulator OMNeT++, and the Veins framework that synchronizes both via the TraCI protocol. SUMO handles the road network and vehicle mobility, while OMNeT++ executes the communication infrastructure logic, and Veins links vehicle dynamics to message flows in real time.

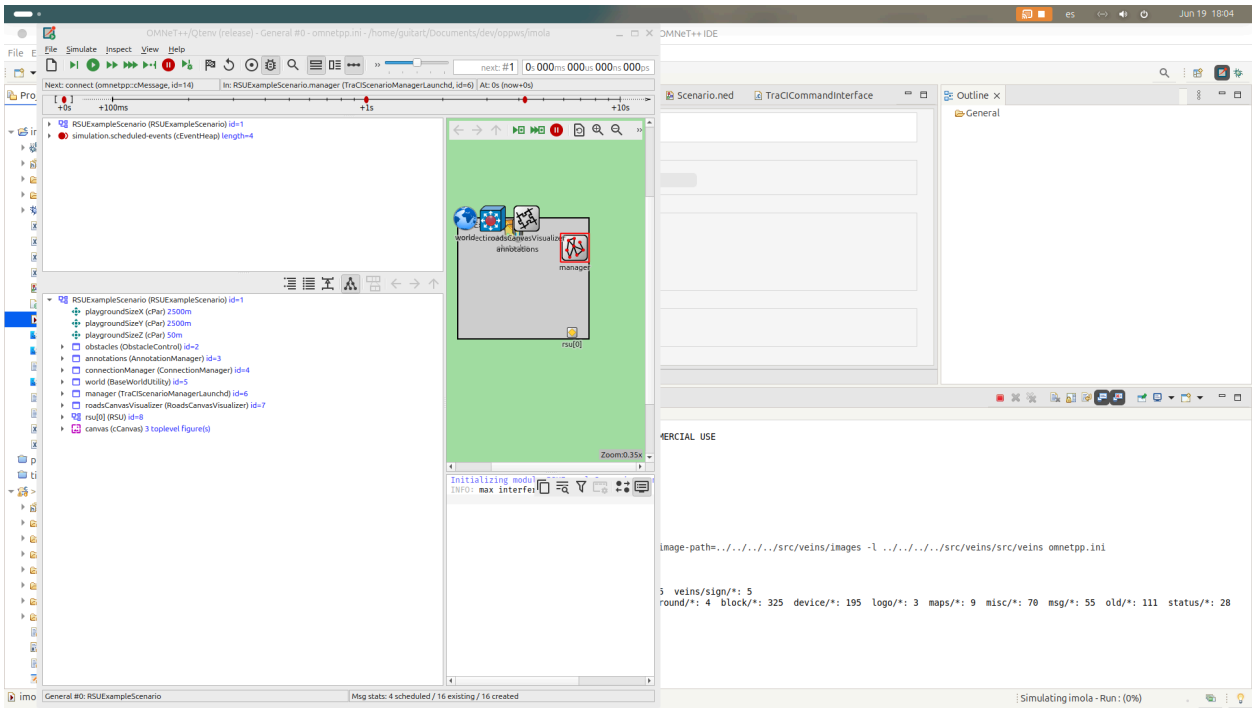

Figure 1 – Overall view of the Imola circuit in SUMO

The digital model of the Imola circuit was generated using SUMO's internal mapping interface, incorporating both the primary racing path and an alternative Pitlane. Routing behavior is defined via *imola.rou.xml*, while conditional rerouting logic is handled through *rerouter.add.xml* using predefined switch points.

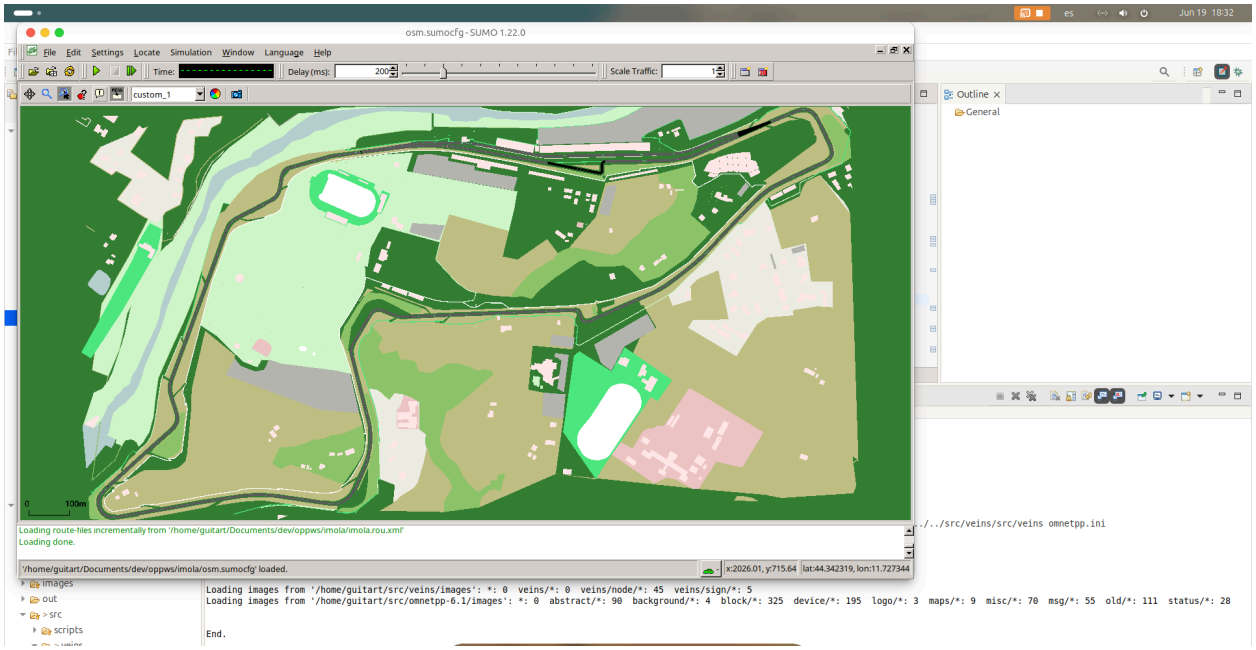

Figure 2 – RaceTrack of Imola mapped for SUMO interface

The RSU was carefully positioned at the center of the circuit map to ensure full coverage and successful delivery of broadcast messages to all vehicles, regardless of their position.

2.2 Pitlane Command Protocol Flow

The "Box Box" protocol is activated through a periodic beacon broadcast, emitted every 5 seconds by the RSU. Each message targets a specific vehicle ID, e.g., “Vehicle 3, box now”. The command is only accepted by the vehicle whose ID matches the one in the message.

Upon correct reception (ID match), the vehicle transitions from yellow (default) to turquoise (active command state) and executes a dynamic reroute to the Pitlane. This behavior is implemented via TraCI message hooks and conditional checks.

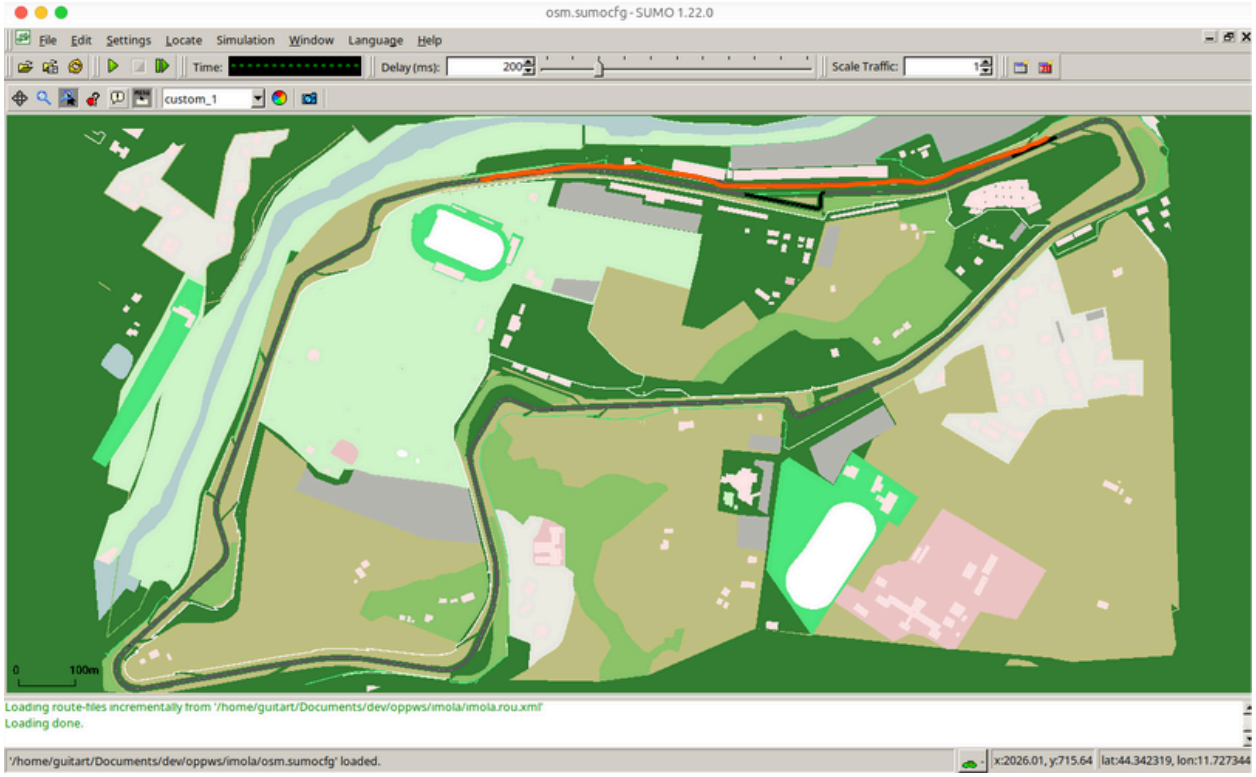

Figure 3 – Highlighted Pitlane route in orange

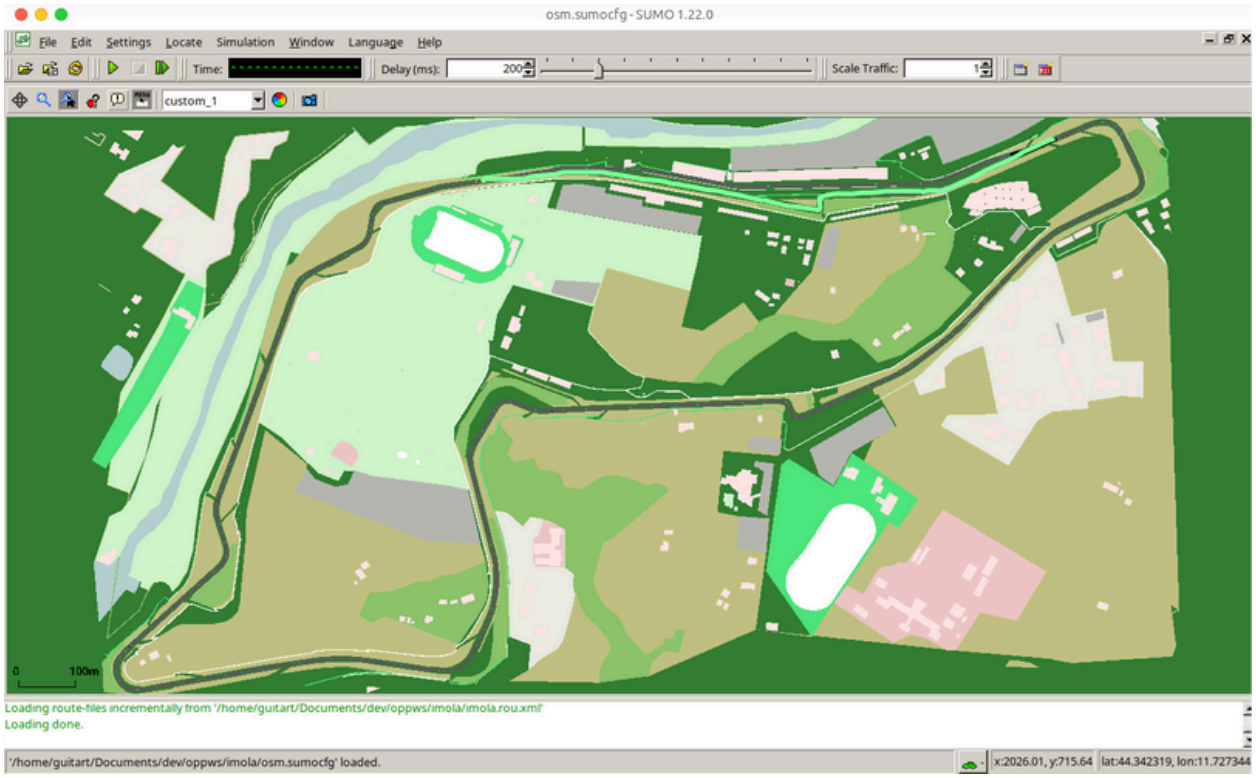

Figure 4 – Highlighted RaceTrack route in green

Redirection points were manually placed near key segments (e.g., entrance of Pitlane) to facilitate the transition. These rerouters check if a valid command has been received and activate the route change.

In addition to routing, a speed constraint is triggered. Vehicles reduce their speed from 300 km/h to 80 km/h upon entering the Pitlane

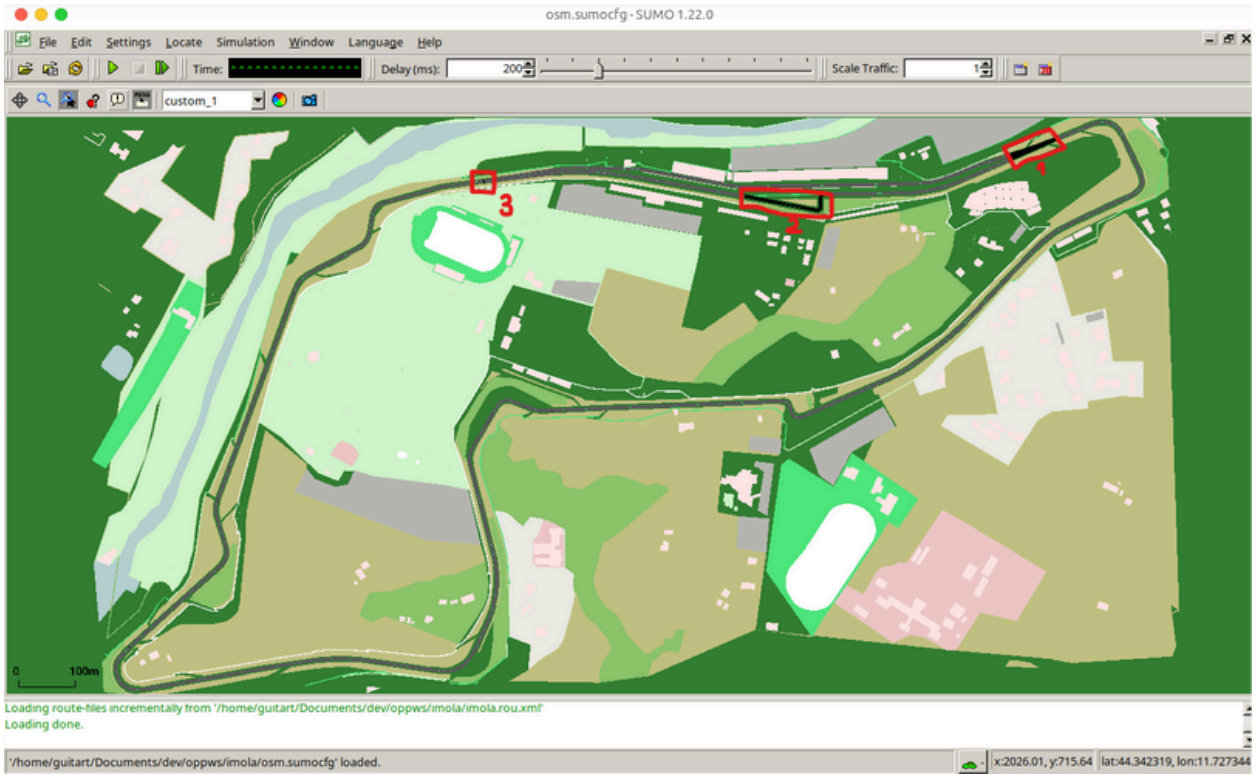

Figure 5 – Redirection points set via *rerouter.add.xml*
(1. Entry Point to Pitlane, 2. Tamburello, 3. Exit of the Pitlane)

2.3 Vehicle FSM (Finite State Machine)

The Vehicle FSM governs how each autonomous car transitions between behavioral states based on message reception and ID validation:

States:

- **RACING** – Default state on the main track.
- **RECEIVED_COMMAND** – The car has received a broadcast, but the ID is not validated.
- **VALID_COMMAND** – The car received a command addressed to its specific ID.
- **REDIRECTED** – The route changes to Pitlane.
- **PITLANE_ENTRY** – Speed is reduced to 80 km/h.

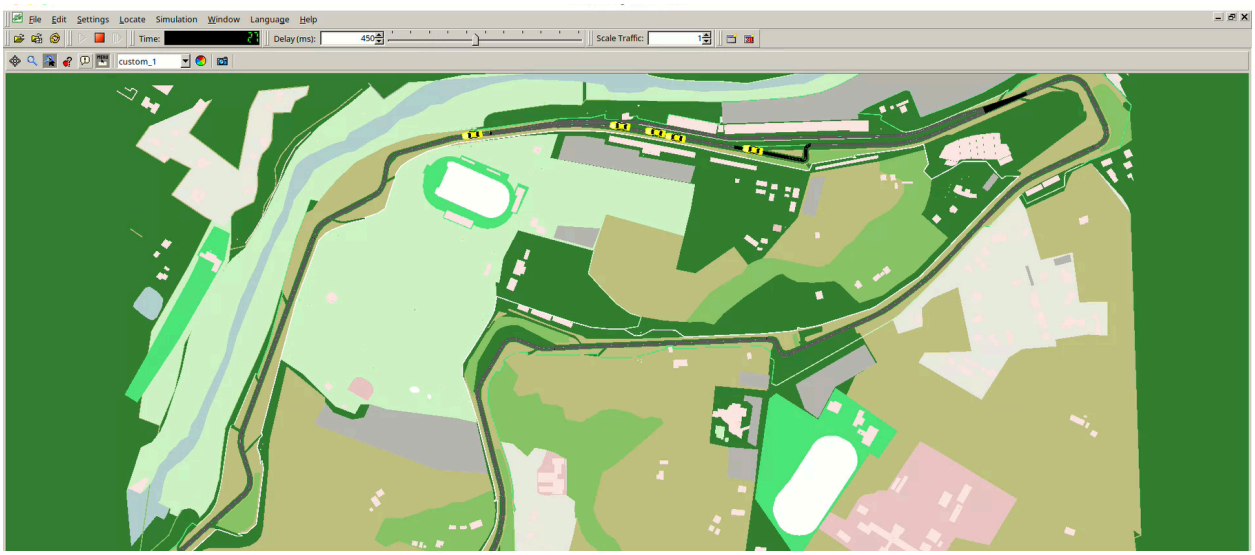

Figure 6 – 5 cars generated racing in the Imola Race Track.

2.4 RSU FSM

The RSU FSM is activated at simulation start and executes a periodic routine:

States:

- **IDLE** – Timer runs in background.
- **SEND_COMMAND** – Every 5 seconds, a message is broadcast to a specific vehicle ID.
- **WAIT** – Until next cycle.

Each broadcast includes a vehicle-specific instruction. Only the car matching the ID will act. The beacon frequency and message format are configured via *omnetpp.ini.*

Together, this system demonstrates selective command activation and controlled rerouting in a racing simulation, ensuring realism, modularity, and full control over scenario variables.

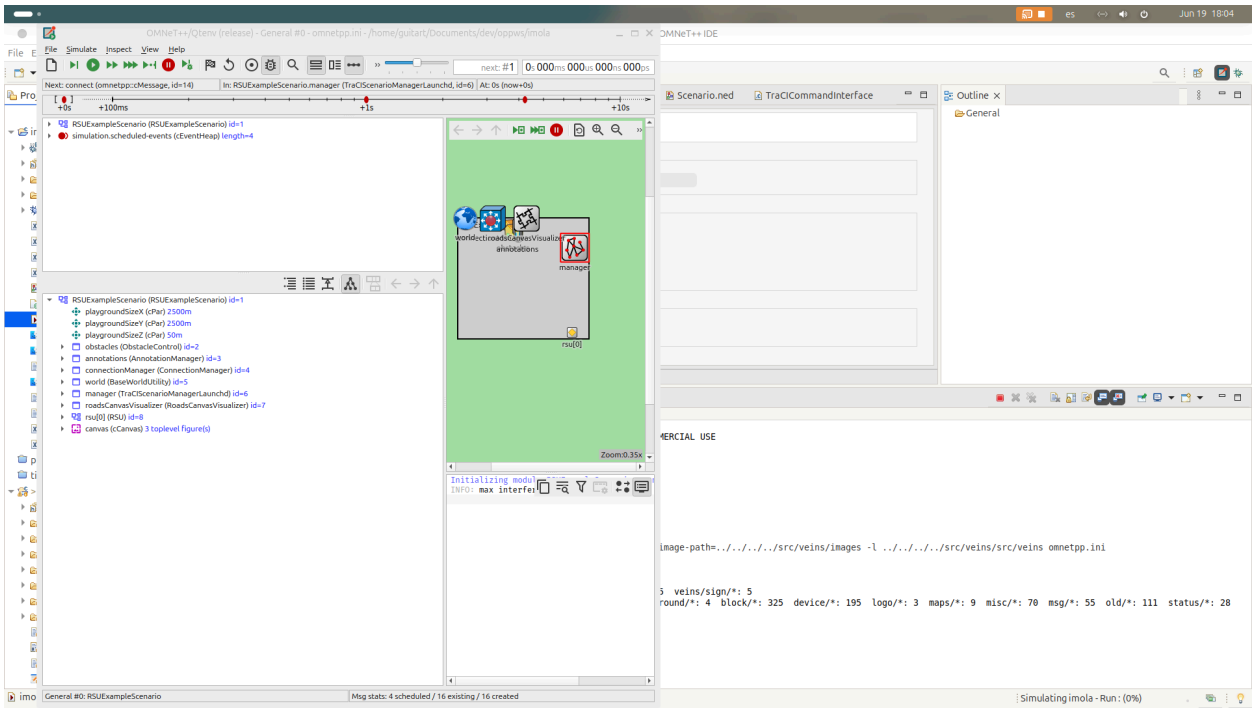

Figure 7 – RSU visual module layout in OMNeT++

3 Execution and Evaluation

This section presents the experimental results derived from the simulation of the "Box Box" protocol on the Imola circuit. The key metrics analyzed focus on the effectiveness of the RSU command, route modification, selective vehicle response, and speed transition upon entering the Pitlane.

3.1 Simulation Configuration

The simulation was conducted using the Imola circuit model configured via SUMO, with five autonomous vehicles operating in a looped race pattern. The RSU was strategically placed at the center of the track to ensure global broadcast coverage and was configured to emit a beacon every 5 seconds via OMNeT++. Each message targeted a unique vehicle ID, simulating individual “Box Box” instructions.

- Vehicles: 5 autonomous cars with IDs from 1 to 5.
- RSU beaconing frequency: every 5 seconds.
- RSU location: central coordinates of the circuit to maximize message reach.
- Activation condition: the vehicle only responds if the received message contains its own ID.
- Speed before Pitlane: up to 300 km/h (no artificial limit).
- Speed in Pitlane: automatically limited to 80 km/h after rerouting.

The system is programmed to change the visual state of the vehicle when one of those RSU messages are received correctly, so this mechanism allows for rapid identification of each vehicle's status during the simulation:

- Yellow Color: Normal state (no command received or invalid).
- Turquoise Color: Valid command received and in execution.

3.2 Command Reception Success Rate

Due to the spatial limitation of the beacon range, the RSU was strategically positioned at the center of the circuit. This ensured effective coverage for all vehicles along their paths. Protocol success is defined by the match between the vehicle ID specified in the message and the vehicle that receives the instruction.

Throughout a typical simulation run of 60 seconds:

- **12 messages** were broadcast by the RSU (one every 5 seconds).
- **5 of those** matched existing vehicle IDs (valid commands).
- **3 vehicles** successfully altered their routes and entered the Pitlane.

This gives a **60% success rate** in command execution, not due to message loss, but due to ID matching variability and the random distribution of vehicle positions. The speed reduction on rerouting occurred seamlessly, demonstrating dynamic behavior adaptation in real time.

Metric	Observed Value
Messages sent by the RSU	12/min (every 5 s)
Commands correctly executed	3 out of 5 vehicles
Success rate	60%
Vehicles Redirected (Sample Run)	Variable (2-3 vehicles typically matched across a full loop)

3.3 Protocol Validation

The observed results confirm that:

- The ID-based validation logic functions correctly.
- The protocol is robust against incorrect or misdirected commands.
- Speed transitions are executed in sync with route changes.

Although no baseline scenario without the protocol was implemented for quantitative comparison, it can be stated that in the absence of the protocol, vehicles would never leave the main track, implying that the rerouting behavior is entirely dependent on RSU activation.

4. Conclusions and Future Work

This project successfully demonstrates a functional V2I rerouting protocol simulating Formula 1-style "Box Box" commands, executed through a controlled vehicular scenario on a digitally reconstructed version of the Imola circuit. The integration of SUMO, OMNeT++, and Veins enabled real-time mobility and communication synchronization between a central RSU and five autonomous vehicles operating in a race loop.

The protocol logic was validated through a finite state machine (FSM) model that governed both vehicle behavior and RSU broadcasting patterns. During simulation, the RSU periodically issued targeted messages to individual vehicle IDs. If the ID matched, the vehicle dynamically altered its route and entered the Pitlane while reducing speed to the regulatory 80 km/h. Visual state transitions (yellow to turquoise) provided real-time feedback on vehicle status.

Results showed a 60% success rate in command execution due to ID-matching logic, not communication failure. Vehicles were able to process route changes dynamically, and speed transitions occurred as expected. The simulation demonstrated the feasibility of conditional rerouting triggered by network-level instructions, confirming the integrity of the protocol's reactive behavior.

However, the current implementation was limited to Pitlane entry only. No re-entry logic was implemented, and no persistent logging was used for post-simulation analysis. Message delivery was assumed reliable under ideal conditions, and no interference or probabilistic loss models were incorporated.

4.1 Future Work

To expand this work into a more complete and realistic vehicular communication protocol, several key improvements are proposed:

- Pitlane Exit and Re-Merge Logic: Implement route re-entry from the Pitlane to the main circuit. This would involve additional FSM transitions and position-aware decision logic to rejoin safely without disrupting traffic flow.

- Protocol Logging and Post-Analysis: Integrate .csv or .xml logging of key metrics such as command timestamps, vehicle states, reroute triggers, and travel times. This would allow more rigorous post-analysis and visual reporting.

- Packet Loss and Redundancy Simulation: Introduce probabilistic message drop rates or simulated interference to evaluate protocol robustness. A retry mechanism or acknowledgment feedback loop could be added for critical messages.

- RSU Placement Strategies: Currently, a single RSU placed at the center of the map ensures coverage. Future scenarios may test distributed RSUs, overlapping broadcast zones, or mobility-triggered RSU handovers to evaluate performance under broader topologies.

- Baseline Comparison: Include a control scenario without the RSU logic, in which vehicles follow only the base route. This would allow quantitative measurement of the protocol's actual impact on behavior, latency, and route flexibility.

- Scalability with More Vehicles and Complex Routes: Extend the simulation to larger fleets and more dynamic environments to test the scalability and responsiveness of the rerouting protocol.

These extensions would not only complete the protocol cycle but also bring the simulation closer to real-world vehicular network scenarios, where reliability, safety, and adaptability are fundamental.

Usage Report

The Imola Pitlane Protocol was developed as a standalone simulation scenario based on the Veins framework, using OMNeT++ and SUMO as the underlying simulators. The protocol replicates the behavior of RSU-initiated rerouting commands in a racing context, requiring a specific configuration to ensure all mobility and communication components operate in sync.

To begin testing the protocol, users must have a compatible simulation stack installed. This includes:

- **OMNeT++** (v6.0 or later), for network simulation and FSM execution.
- **SUMO** (v1.14.1 or later), for vehicular mobility and route dynamics.
- **Veins**, which acts as a synchronization bridge between the two simulators via the TraCI protocol.

All tools should be installed according to their official documentation. I recommend setting up the environment in a Unix-based system and organizing both Veins and this project under the same workspace directory for compatibility.

The ImolaProject repository contains all custom configurations, source files, and SUMO network definitions required to replicate our scenario. Once cloned, the project should be imported into OMNeT++ alongside Veins:

File -> Import -> Existing Projects into Workspace

Upon import, verify that Veins is correctly referenced as a dependency within the project settings. The main configuration file is *omnetpp.ini*, which orchestrates simulation parameters including vehicle count, RSU beacon frequency, routing logic, and GUI behavior.

To execute the simulation:

1. Compile the project with OMNeT++ build tools.
2. Launch the simulation using Qtenv for visual tracking or Cmdenv for performance analysis.
3. Ensure that SUMO is started in GUI mode to monitor vehicle color transitions and route changes.

Vehicles in this simulation behave autonomously unless receiving a valid "Box Box" instruction from the RSU. Only vehicles whose ID matches the broadcast message will change their route and reduce speed upon Pitlane entry.

This setup allows the protocol to be evaluated under controlled conditions, highlighting the selective behavior and responsiveness of vehicles based on ID-filtered RSU messages. The current version does not include automatic logging to CSV or external files, but future iterations may incorporate such features for quantitative post-analysis.