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

Index 2
1. Project Description 3
1.1 Project Objectives 4
2. Protocol Design 5
2.1 System Architecture Overview 5
2.2 Pitlane Command Protocol Flow 7
2.3 Vehicle FSM (Finite State Machine) 9
2.4 RSU FSM 10
3. Execution and Evaluation 13
3.1 Simulation Configuration 13
Pitlane Entry vs. Normal Lap – Time-Distance Impact Analysis 14
3.2 Command Reception Success Rate 16
Sensitivity Analysis of RSU Beacon Interval 17
3.3 Protocol Validation 19
4. Conclusions and Future Work 20
Usage Report 22
Repository Access 23
References 24

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:

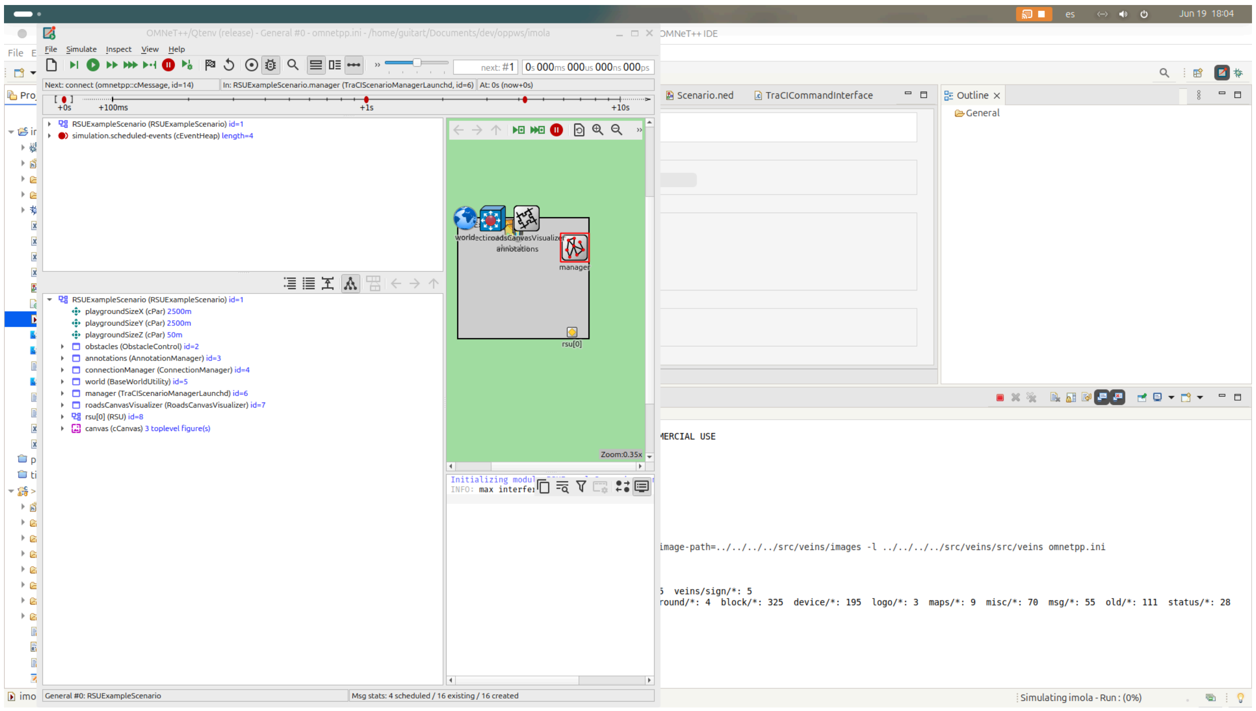
- Design a robust RSU-triggered route-changing protocol using SUMO and OMNeT++.
- Define a message flow enabling autonomous vehicles to receive and act on external instructions dynamically.
- Implement FSMs to manage both infrastructure and vehicle behavior in response to timed or triggered events.
- 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.



```
Next: connect (omnetpp::cMessage, id=14)
In: RSUExampleScenario.manager (TraCIScenarioManagerLaunchd, id=6)
At: 0s (now+0s)
```

```
RSUExampleScenario (RSUExampleScenario) id=1
simulation.scheduled-events (cEventHeap) length=4
```

```
RSUExampleScenario (RSUExampleScenario) id=1
    playgroundSizeX (cPar) 2500m
    playgroundSizeY (cPar) 2500m
    playgroundSizeZ (cPar) 50m
    obstacles (ObstacleControl) id=2
    annotations (AnnotationManager) id=3
    connectionManager (ConnectionManager) id=4
    world (BaseWorldUtility) id=5
    manager (TraCIScenarioManagerLaunchd) id=6
    roadsCanvasVisualizer (RoadsCanvasVisualizer) id=7
    rsu[0] (RSU) id=8
    canvas (cCanvas) 3 toplevel figure(s)
```

```
Initializing modul
INFO: max interfe
```

```
MERCIAL USE

image-path=../../../../src/veins/images -l ../../../../src/veins/src/veins omnetpp.ini

5  veins/sign/*: 5
round/*: 4  block/*: 325  device/*: 195  logo/*: 3  maps/*: 9  misc/*: 70  msg/*: 55  old/*: 111  status/*: 28
```

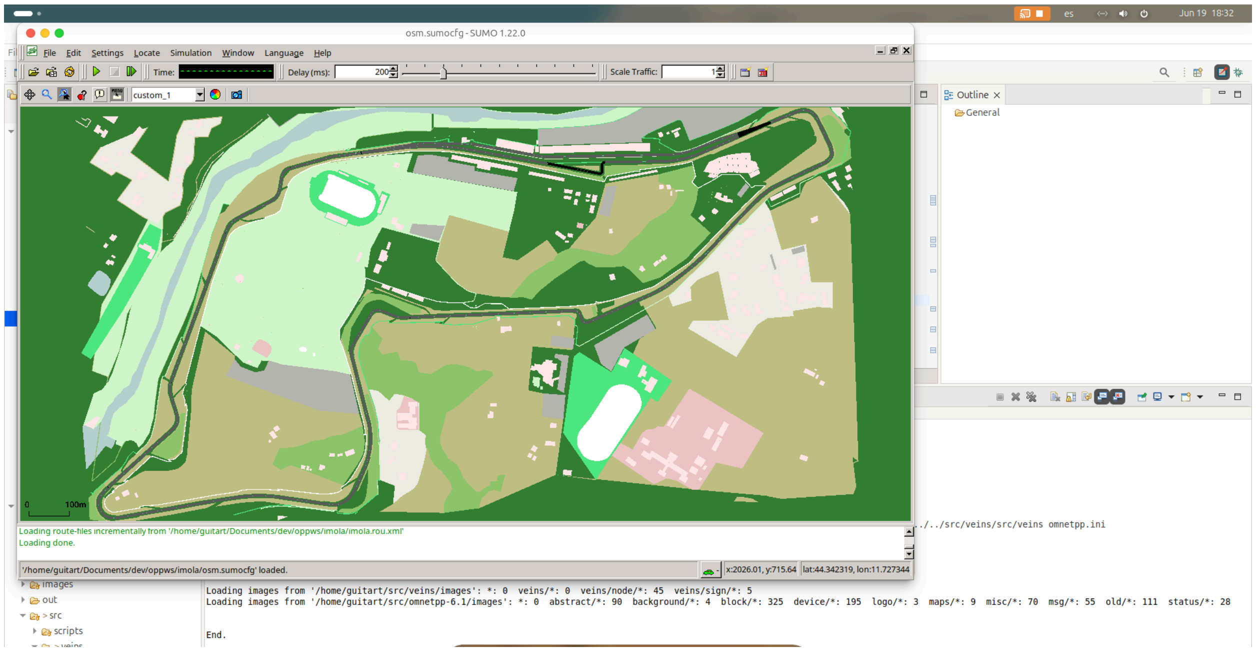
```
General #0: RSUExampleScenario
Msg stats: 4 scheduled / 16 existing / 16 created
Simulating imola - Run : (0%)
```

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.

The Vehicle Targeting Mechanism consists in the following process, the RSU emits a broadcast beacon every *T* seconds containing a BoxBox(ID) message, where the target ID is randomly selected among the active vehicles (e.g., [1, 5]). Each vehicle receiving the message compares the embedded ID with its own. If the match occurs, the vehicle enters the Pitlane maneuver.

Currently, the ID is selected through a basic random generator. However, this approach may result in low success rates due to mismatches.



```
Loading route-files incrementally from '/home/guitart/Documents/dev/oppws/imola/imola.rou.xml'
Loading done.
'/home/guitart/Documents/dev/oppws/imola/osm.sumocfg' loaded.
x:2026.01, y:715.64   lat:44.342319, lon:11.727344
```

```
../../src/veins/src/veins omnetpp.ini
Loading images from '/home/guitart/src/veins/images': *: 0  veins/*: 0  veins/node/*: 45  veins/sign/*: 5
Loading images from '/home/guitart/src/omnetpp-6.1/images': *: 0  abstract/*: 90  background/*: 4  block/*: 325  device/*: 195  logo/*: 3  maps/*: 9  misc/*: 70  msg/*: 55  old/*: 111  status/*: 28


End.
```

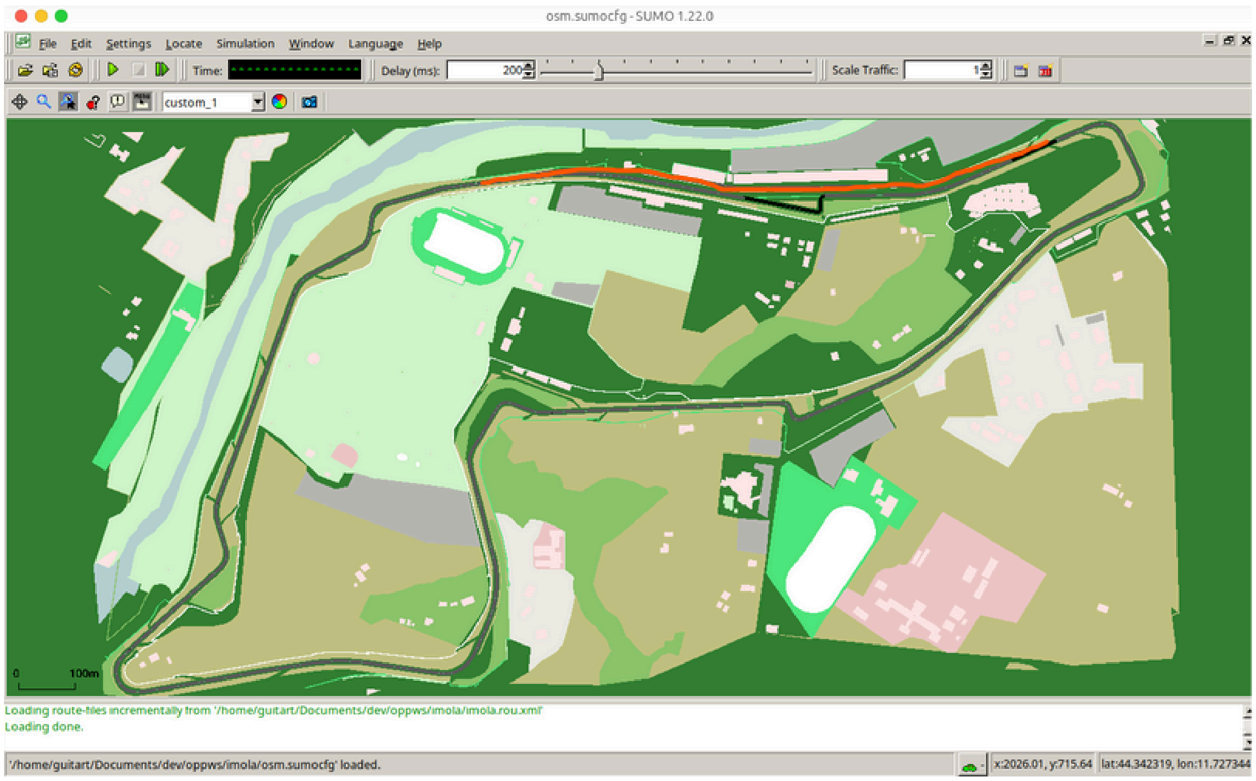
Figure 2 – RaceTrack of Imola mapped for SUMO interface

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.

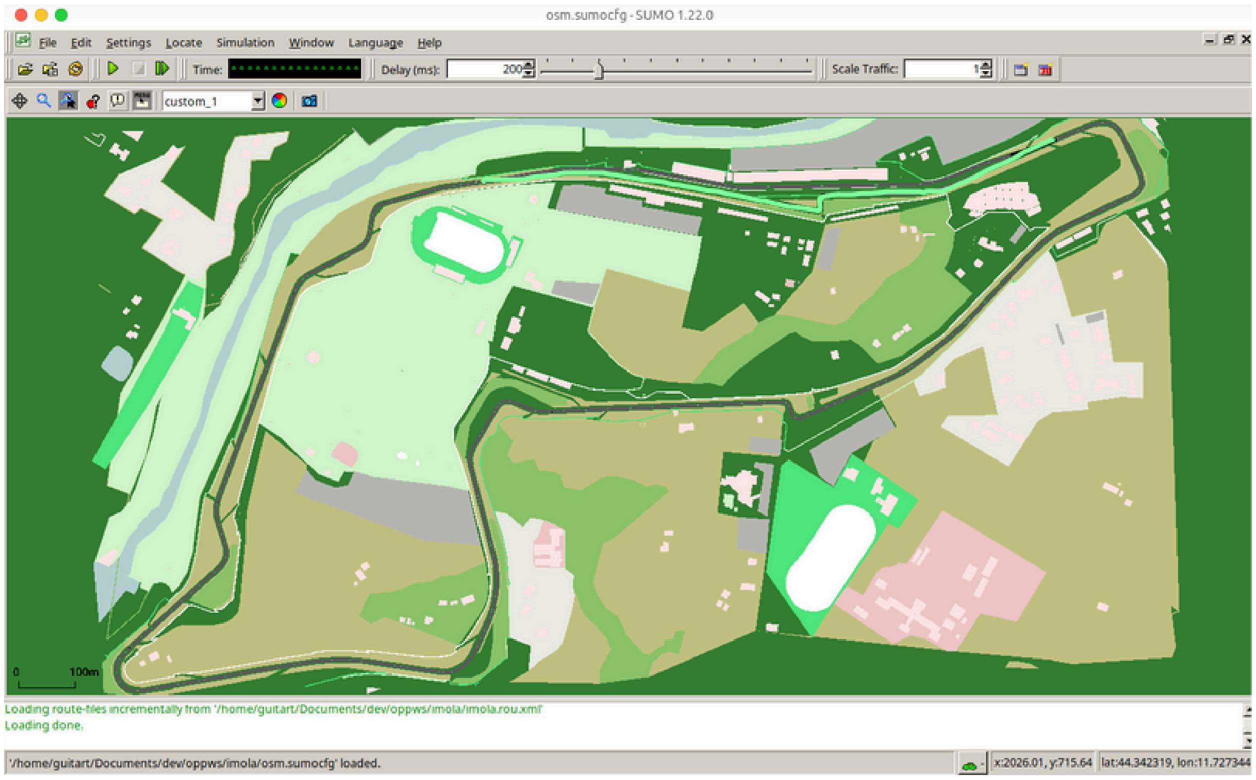
Upon correct reception (ID match), the vehicle transitions from yellow (default) to turquoise (active command state) and executes a dynamic reroute to the Pitlane. This behavior is implemented via TraCI message hooks and conditional checks.



```
Loading route-files incrementally from '/home/guitart/Documents/dev/oppws/imola/imola.rou.xml'
Loading done.
```

```
'/home/guitart/Documents/dev/oppws/imola/osm.sumocfg' loaded.    x:2026.01, y:715.64    lat:44.342319, lon:11.727344
```

Figure 3 – Highlighted Pitlane route in orange

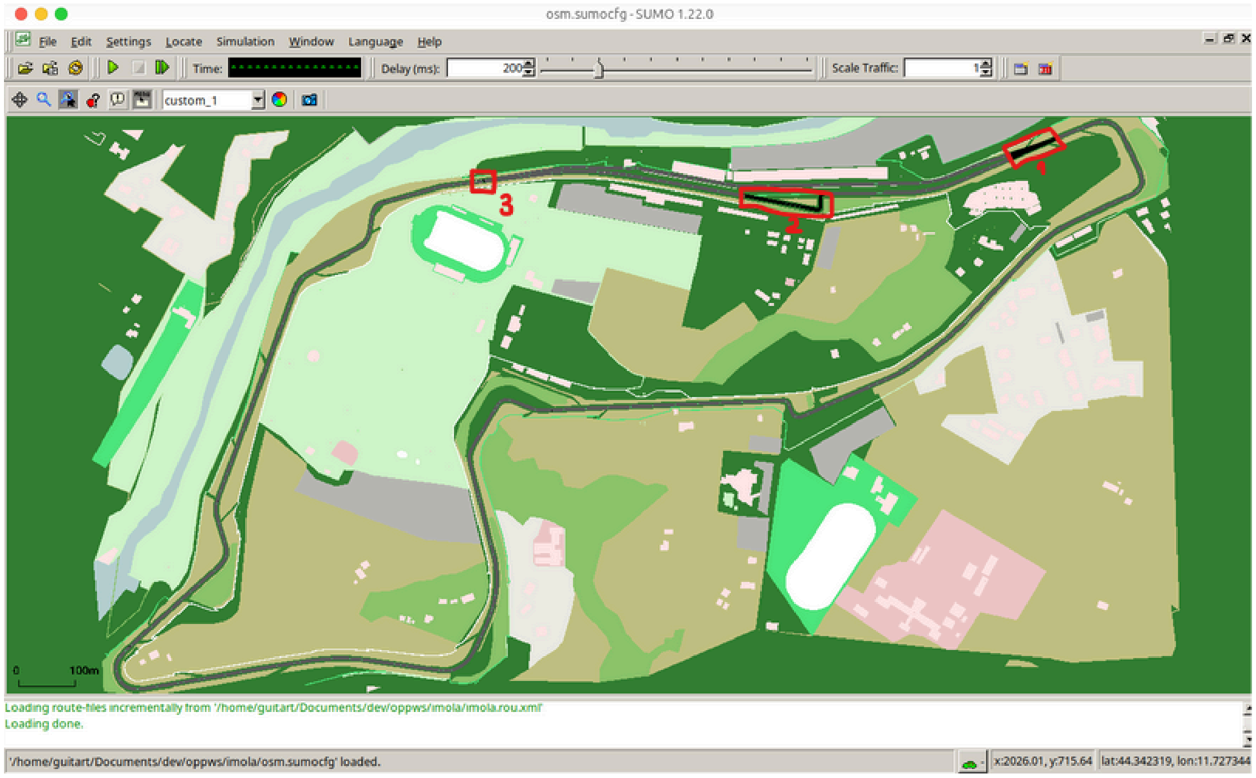


```
Loading route-files incrementally from '/home/guitart/Documents/dev/oppws/imola/imola.rou.xml'
Loading done.
'/home/guitart/Documents/dev/oppws/imola/osm.sumocfg' loaded.
x:2026.01, y:715.64
lat:44.342319, lon:11.727344
```

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



```
Loading route-files incrementally from '/home/guitart/Documents/dev/oppws/imola/imola.rou.xml'
Loading done.
'/home/guitart/Documents/dev/oppws/imola/osm.sumocfg' loaded.
```

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.

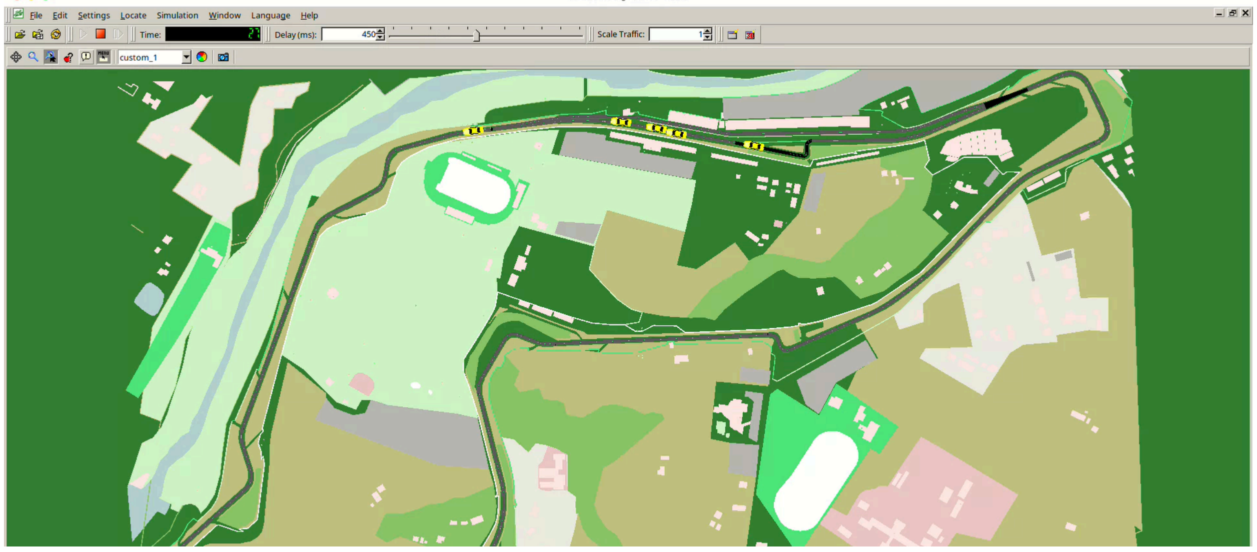


Figure 6 – 5 cars generated racing in the Imola Race Track.

2.4 RSU FSM

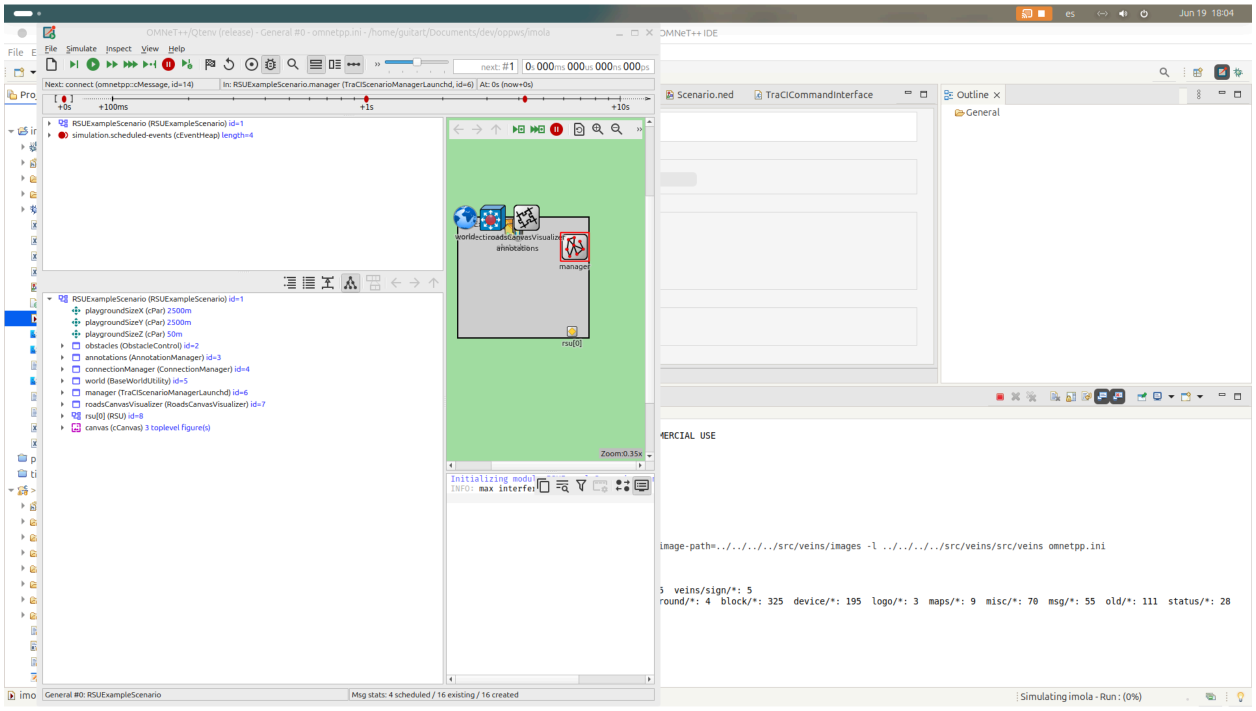
The RSU FSM is activated at simulation start and executes a periodic routine:

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.



```
RSUExampleScenario (RSUExampleScenario) id=1
    playgroundSizeX (cPar) 2500m
    playgroundSizeY (cPar) 2500m
    playgroundSizeZ (cPar) 50m
    obstacles (ObstacleControl) id=2
    annotations (AnnotationManager) id=3
    connectionManager (ConnectionManager) id=4
    world (BaseWorldUtility) id=5
    manager (TraCIScenarioManagerLaunchd) id=6
    roadsCanvasVisualizer (RoadsCanvasVisualizer) id=7
    rsu[0] (RSU) id=8
    canvas (cCanvas) 3 toplevel figure(s)
```

```
MERCIAL USE

image-path=../../../../src/veins/images -l ../../../../src/veins/src/veins omnetpp.ini

5  veins/sign/*: 5
round/*: 4  block/*: 325  device/*: 195  logo/*: 3  maps/*: 9  misc/*: 70  msg/*: 55  old/*: 111  status/*: 28
```

OMNeT++/Qtenv (release) - General #0 - omnetpp.ini - /home/guitart/Documents/dev/oppws/imola
File
Simulate
Inspect
View
Help
Next: connect (omnetpp::cMessage, id=14)
In: RSUExampleScenario.manager (TraCIScenarioManagerLaunchd, id=6)
At: 0s (now+0s)
simulation.scheduled-events (cEventHeap) length=4
world
annotations
manager
rsu[0]
Zoom:0.35x
Scenario.ned
TraCICommandInterface
Outline
General
OMNeT++ IDE
General #0: RSUExampleScenario
Msg stats: 4 scheduled / 16 existing / 16 created
Simulating imola - Run : (0%)

Figure 7 – RSU visual module layout in OMNeT++

To improve clarity and completeness, now I include a formal FSM (Finite State Machine) diagram representing the internal behavior of each vehicle in response to RSU beacons. This complements the textual logic previously described and clarifies the state transitions involved in the protocol execution.

Each vehicle starts in the **"Cruising"** state, maintaining constant speed. Upon receiving a **BoxBox(ID)** message from the RSU, it evaluates the message content. If the target ID matches its own, the vehicle transitions to **"Routing to Pitlane"**, activates rerouting logic, and decelerates to 80 km/h. Once the Pitlane section is completed, the vehicle enters a transient **"Merging back"** state before returning to **"Cruising"**, resuming its normal trajectory and speed.

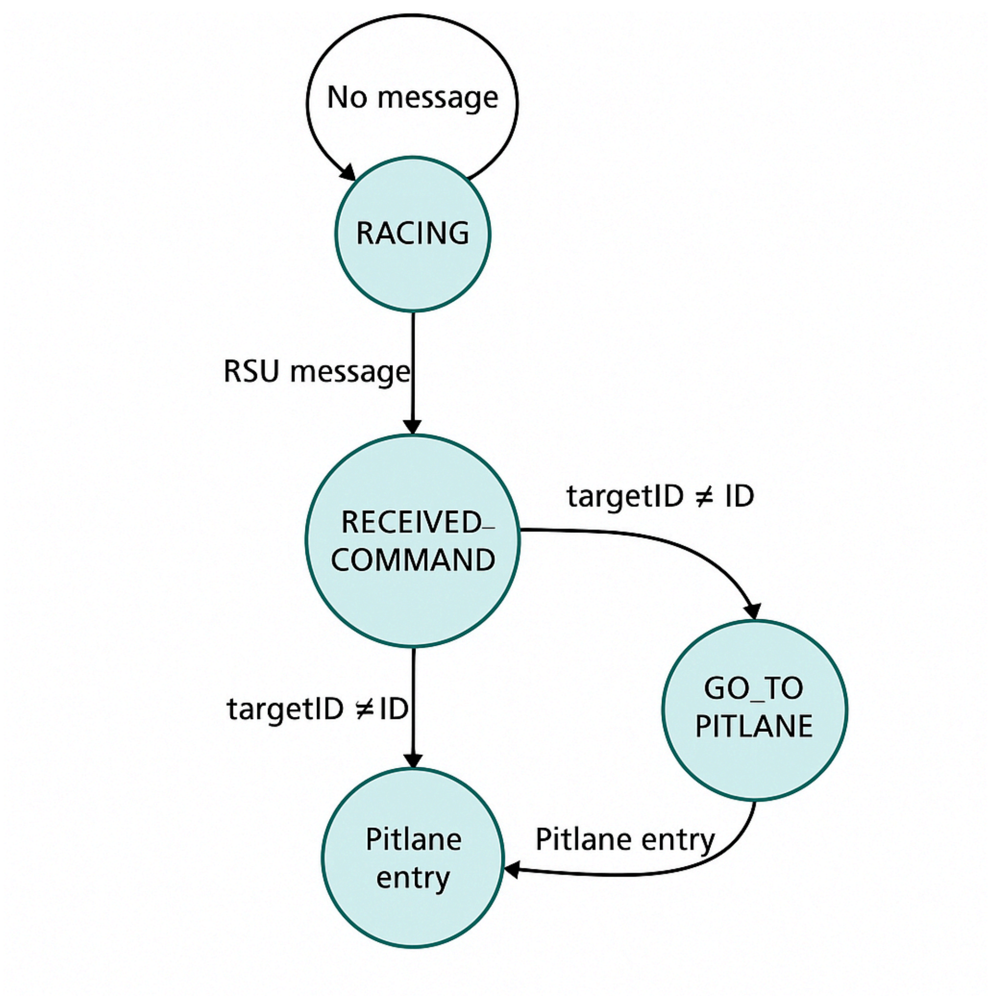


Figure 8 – Finite State Machine (FSM) diagram describing vehicle behavior within the Pitlane protocol FSM

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.

In Formula 1, pitlane maneuvers significantly affect lap times due to strict speed limits and necessary transitions between race speed and pitlane speed. This section presents a quantitative breakdown of the entering pitlane maneuver, braking entry from full-speed to pitlane speed limit.

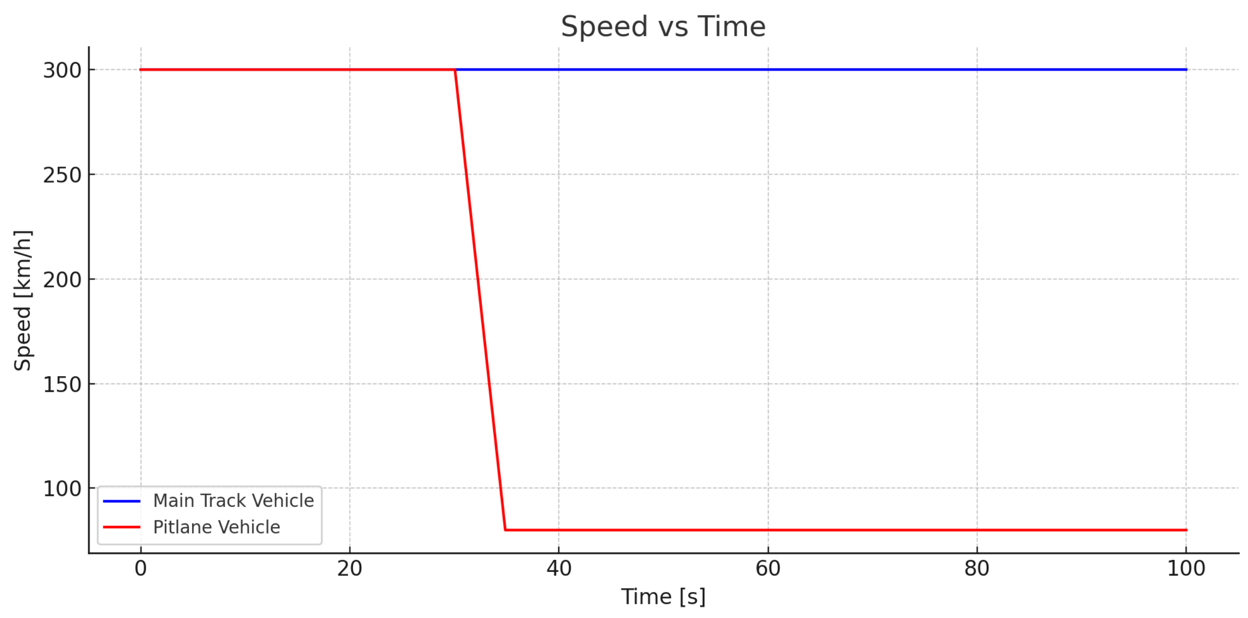


Figure 9 – Speed vs Time plot comparison between normal route vs deceleration of speed entering to the pitlane once is rerouted.

Pitlane Entry vs. Normal Lap – Time-Distance Impact Analysis

To quantitatively support the speed transition illustrated in *Figure 8* and compare it against a normal uninterrupted racing lap, *Figure 9* overlays both trajectories in terms of velocity over time. The red curve represents the complete pitlane maneuver, including realistic braking and acceleration phases, while the blue dashed line indicates the uninterrupted lap at constant racing speed (300 km/h or 83.33 m/s).

The maneuver is segmented into three key phases, each calculated analytically below:

Braking Phase

Initial Speed = 83.33m/s

Final Speed = 22.22m/s

Duration = 1.4s

- Deceleration:

$ab = tbvf - vi = 1.422.22 - 83.33 =- 43.65m/s2$

- Distance covered:

$db = 2vi + vf\cdot tb = 52.78\cdot 1.4\approx 73.9m$

<u>Acceleration Phase</u>

From 22.22m/s to 83.33m/s

Duration = 3.4s

- Acceleration:

$aa = tavf - vi = 3.461.11\approx 17.97m/s2$

Distance to reach top speed:

$da = 2vi + vf\cdot ta = 52.78\cdot 3.4\approx 179.5m$

- Deceleration:

$ab = tbvf - vi = 1.422.22 - 83.33 =- 43.65m/s2$

Distance covered:

$db = 2vi + vf\cdot tb = 52.78\cdot 1.4\approx 73.9m$

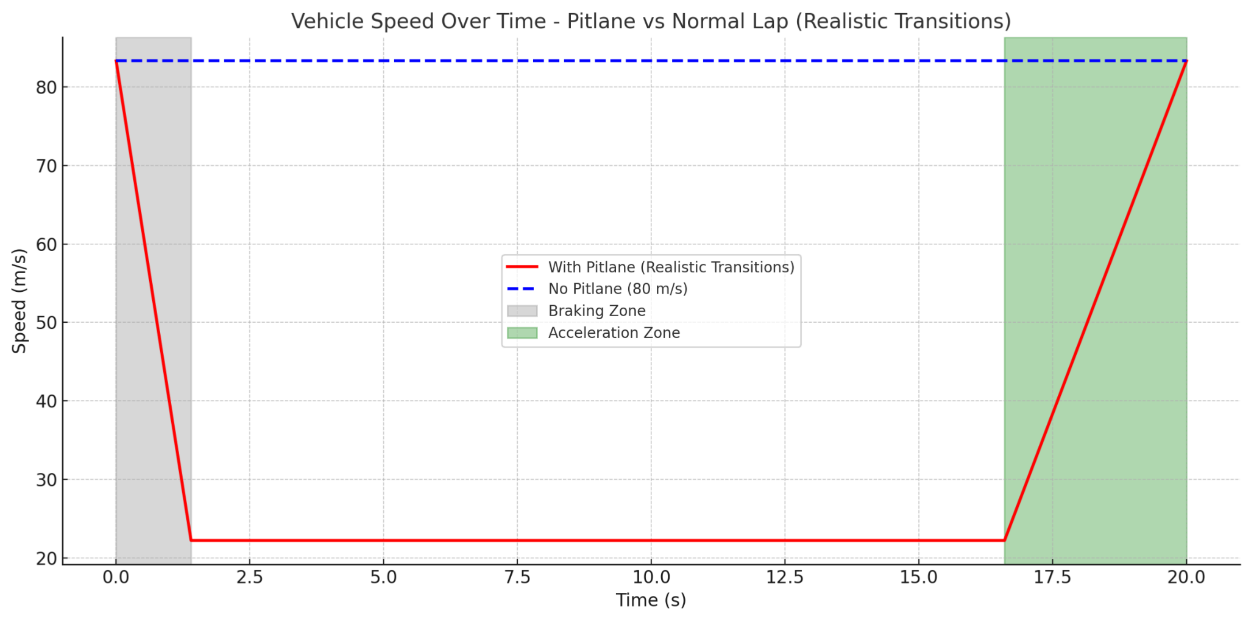


Figure 10 – Vehicle speed over the time plot comparison between normal lap vs pitlane lap with the zones and specific times with breaking and acceleration calculations.

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)

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.

Sensitivity Analysis of RSU Beacon Interval

To evaluate the impact of RSU beacon frequency on the effectiveness of the BoxBox protocol, we perform a sensitivity analysis by varying the beacon emission interval. Since RSU messages are broadcast and contain a specific vehicle ID, the frequency at which these are sent directly influences the probability that a vehicle within range receives and matches a valid command.

- RSU has a fixed communication range
- Only vehicles that receive a message **while in range** *and* **with a matching ID** can respond to the command.

To assess the responsiveness and robustness of the proposed V2I Pitlane protocol, we conducted a sensitivity analysis over the RSU beacon interval, i.e., the time between consecutive messages broadcasted by the RSU.

The current implementation uses a **5-second interval**, which under normal lap conditions (average lap duration ~74 seconds) yields approximately **15 messages per lap**. Each message carries a command targeted at a random vehicle ID from the pool of 5 active vehicles. Thus, the probability of issuing a valid BoxBox command to the *correct vehicle* at each opportunity is 1 in 5 (20%).

We extend this analysis resetting the interval implementation to shorter and longer beacon intervals, from **0.25s to 20s**, and check the success rate—defined as the probability that each vehicle receives at least one matching BoxBox command during a full lap.

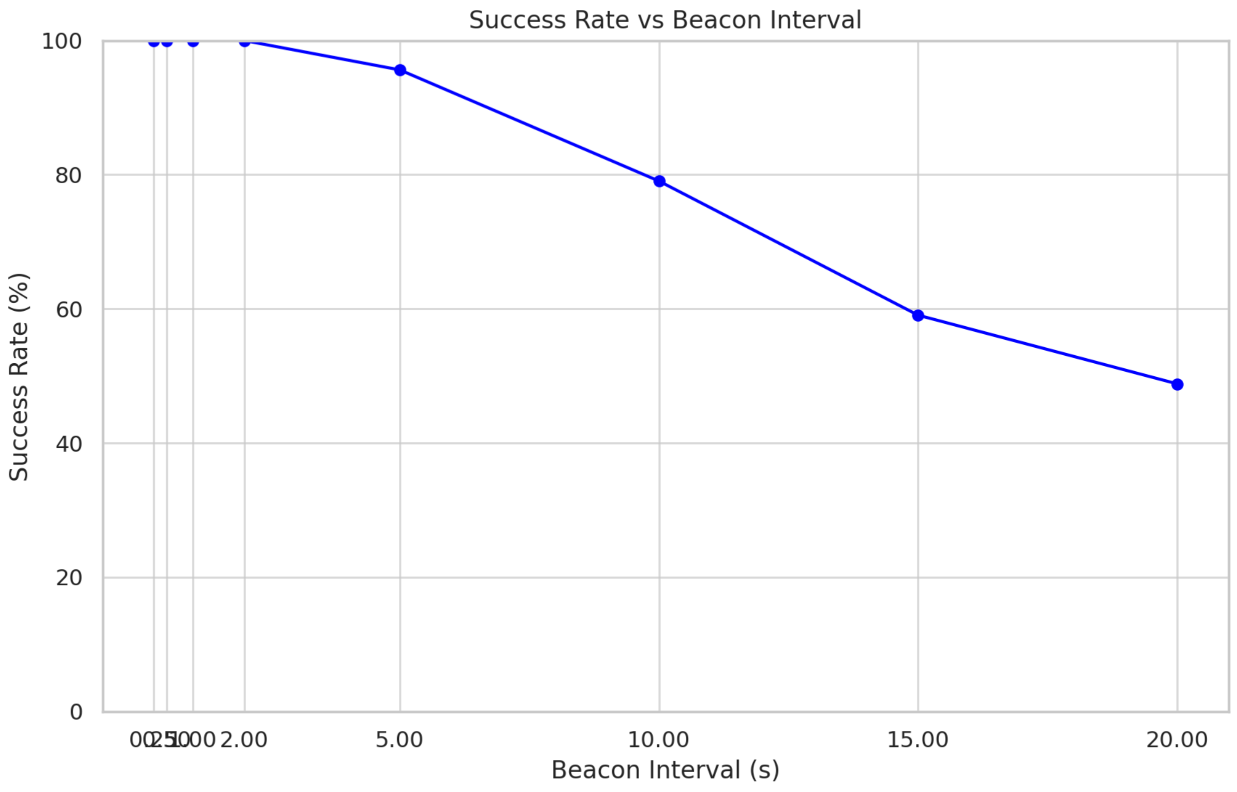


Figure 11 – Vehicle speed over the time plot comparison between normal lap vs pitlane lap with the zones and specific times with breaking and acceleration calculations.

- Intervals ≤ 2s guarantee almost complete reliability (≥99.9%) for successful command delivery within a lap.

- The default 5s setting maintains a high but not full reliability at ~95%.

- For intervals above 10s, the likelihood of successful vehicle rerouting drops significantly below 80%, which could compromise the simulation objectives.

- Thus, fine-tuning the beacon interval is critical: lowering it improves delivery success, but at the cost of increased message frequency and potential wireless channel congestion in denser deployments.

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 designed and validated a command-based rerouting protocol in a simulated Vehicle-to-Infrastructure environment modeled on the Imola Formula 1 circuit. By leveraging SUMO, OMNeT++ and the Veins framework, we replicated a scenario in which a centralized roadside unit (RSU) periodically broadcasts selective “Box Box” commands, prompting specific vehicles to deviate from the main racing trajectory and enter the Pitlane.

From a technical standpoint, the implemented FSMs and command validation logic correctly manage conditional rerouting and vehicle state transitions. Vehicles reduce their speed and alter their path only when the received message matches their unique ID, ensuring a deterministic response model. Visual cues (vehicle color changes) further assist in verifying system behavior during the simulation runtime.

However, several **limitations inherent to the current implementation** must be acknowledged:

- **No Pitlane re-entry**: Once a vehicle enters the Pitlane, it does not reintegrate into the racing circuit. This restricts the simulation to one-way transitions, which simplifies state handling but limits real-world applicability.

- **Ideal communication model**: The RSU assumes perfect broadcast coverage with no latency or packet loss. This abstracts away the inherent unreliability of V2X communication, particularly in dynamic or obstructed environments.

- **Lack of feedback or acknowledgment**: The system does not include confirmation messages from vehicles to the RSU. This makes the command execution success rate observable only via indirect simulation metrics, not protocol-level confirmation.

- **Single RSU model**: The entire circuit is managed by one RSU placed at the map center. While effective in simulation, real-world deployments would likely require overlapping RSUs and handovers, especially in larger or more obstructed areas

- **Why Messages Fail:**
  Several factors contribute to the failure of BoxBox commands:

- The beacon is sent only once, without retries or acknowledgments.
- The vehicle receiving the message does not match the target ID.
- The RSU is not within communication range of the intended vehicle..

Despite these simplifications, the project achieves a **proof-of-concept milestone** for selective, ID-based rerouting via broadcast beacons. The RSU successfully initiates differentiated behavior across vehicles using only public communication, validating the concept of external behavioral overrides in autonomous systems.

From a practical perspective, this work has direct implications for **intelligent traffic management systems**, **safety corridors**, or **autonomous vehicle testing protocols**. Being able to dynamically reroute vehicles based on external commands could be valuable for emergency control, infrastructure-aware driving, or testing compliance scenarios.

Usage Report

The Imola Pitlane Protocol was developed as a standalone simulation scenario based on the Veins framework, integrating **OMNeT++**, **SUMO**, and **TraCI**. It replicates a realistic V2I interaction model where an RSU periodically broadcasts rerouting commands ("Box Box") to specific vehicles on the Imola circuit.

To run the simulation, users must have:

- **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.

Key Files and Functions

- omnetpp.ini: central configuration file. Defines:
    - Simulation length and GUI preferences.
    - RSU beacon interval (5s).
    - Color change triggers (yellow to turquoise).
    - Vehicle and network module references.
- rerouter.add.xml: specifies three redirection checkpoints:
    - Entry to the Pitlane.
    - Tamburello zone. (Race Path)

- ImolaScenario.ned: OMNeT++ network topology file.
    - Declares RSU and vehicle modules.
    - Sets channel types and spatial layout.
    - Links to SUMO network file (osm.net.xml.gz).

Running the Simulation

1. Import the imola folder as a project in OMNeT++.
2. Build the project via OMNeT++ IDE or make from terminal.
3. Run *imola.launchd.xml* using Qtenv for graphical output.
4. Observe SUMO GUI:
    - Vehicles race around the circuit.
    - On correct ID-match, rerouted vehicles enter Pitlane and change color.

Only vehicles whose ID matches the broadcast message will respond. This highlights the filtered reception behavior and command-driven route switch. By design, vehicles do not rejoin the track after Pitlane entry — this remains open for future integration.

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.

Repository Access

The full simulation environment, source code, and example configuration files are available at: https://github.com/MarcGuitart/ImolaProject

References

1. SUMO Documentation - https://sumo.dlr.de/docs/
2. OMNeT++ - https://omnetpp.org/
3. Veins Framework - https://veins.car2x.org/
4. Veins GitHub Repository - https://github.com/sommer/veins