

# Month 5: Modeling Real-World Systems (Autonomous Vehicles) with UPPAAL-SMC

**Title:** Modeling Real-World Systems (Autonomous Vehicles) with UPPAAL-SMC

## Introduction

In this month, our focus was on applying UPPAAL-SMC to **real-world systems** with a special emphasis on **autonomous vehicles** and **traffic management**. Intersections are among the most critical points in urban traffic systems, where concurrency, timing, and probabilistic behaviors play an essential role. Therefore, our case study involved modeling an **intelligent traffic light system** and autonomous vehicles that interact with it.

The main objective was to:

1. Build a correct UPPAAL-SMC model of an intersection with autonomous vehicles.
2. Introduce **probabilistic elements** (e.g., random arrival of cars, stochastic failures of the traffic light).
3. Verify the model against **safety and liveness properties** using UPPAAL-SMC queries.
4. Demonstrate how real-world decision-making can be analyzed using formal methods.

---

## Week 1: Introduction to UPPAAL Applications in Autonomous Vehicles

- We studied the relevance of **formal modeling** for traffic systems.
- Autonomous vehicles require strict guarantees to avoid unsafe situations (e.g., collisions, deadlocks).
- UPPAAL-SMC was introduced as a tool that supports:
  - **Timed Automata** (to model delays and green/red light timing).
  - **Stochastic behaviors** (exponential distributions for car arrivals, traffic light failures).

- **Model checking queries** (to verify correctness under probabilistic assumptions).
  - We defined the high-level requirements for our model:
    - Cars must eventually cross the intersection (no starvation).
    - The traffic light should cycle between states fairly.
    - Deadlocks must not occur (the system should always progress).
- 

## Week 2: Modeling Autonomous Vehicle Control Systems

- We designed two car templates: **CarA** and **CarB**.
  - States: *Idle*, *Approaching*, *Waiting*, *Crossing*, and *Exited*.
  - Synchronization channels:
    - `car_here!` – signal that a car has arrived.
    - `green?` – wait for green light permission.
    - `passed!` – notify that a car has crossed.
  - Variables:
    - `queue` – number of cars waiting.
    - `crossing_cars` – number of cars currently crossing.
    - `ready_to_cross` – number of cars ready to move on green.
- We also developed the **TrafficLight** template with the following states:
  - *Red* → *Checking* → *Green* → *WaitingToTurnRed* → *Red*.

- **Broken** state added to simulate failures.
  - Guards and assignments were carefully introduced to ensure correct transitions:
    - Example: `queue > 0 && intersection_busy == false && crossing_cars < 2`.
    - These conditions prevent unsafe situations such as two cars colliding.
- 

### Week 3: Simulating System Decisions in Intersections and Emergency Situations

- Various **scenarios** were modeled:
  - Multiple cars are arriving simultaneously.
  - The traffic light turns green when there are no cars waiting.
  - A car is crossing while another is still waiting.
  - Failure and recovery of the traffic light.
- **Exponential rates** were assigned to transitions to simulate realistic probabilistic timings (e.g., car arrival rate = 1, traffic light failure rate = 0.01).
- We introduced `intersection_busy` to prevent unsafe overlaps in decision-making.
- Key SMC queries were designed and tested:
  - `A[] not deadlock` – checked that no deadlocks occur.
  - `Pr[≤100](↯ CarA.Exited && CarB.Exited)` – probability that both cars exit within a time bound.
  - `Pr[≤200](↯ (crossing_cars == 0))` – probability that the intersection eventually clears.
  - Queries combining probabilities with confidence intervals to measure system reliability.

---

## Week 4: Final Project – Modeling an Autonomous Vehicle System

The full model integrated **Cars + Traffic Light** into a single **system declaration**:

```
system TrafficLight, CarA, CarB;
```

- 
- Additional enhancements:
  - Added `passed_count` to measure how many cars successfully crossed.
  - Used `intersection_busy` as a boolean guard to prevent inconsistencies.
  - Ensured that after each crossing, `queue`, `crossing_cars`, and `ready_to_cross` were updated correctly.
- Results from SMC queries:
  - **Deadlock freedom** was achieved.
  - The probability of both cars eventually crossing within a time bound was consistently **high (>0.9)**.
  - The system performance was validated under different simulation runs, confirming reliability.
- This project demonstrated how **formal verification** ensures safety in real-world autonomous systems where concurrency and randomness coexist.

---

## Conclusion

By the end of Month 5, we successfully:

1. Built a **complete UPPAAL-SMC model** for an intelligent traffic light and autonomous cars.
2. Introduced stochastic behaviors to reflect **real-world uncertainties** (e.g., random arrivals, light failures).

3. Verified the system with **safety and liveness queries** to ensure correctness.
4. Produced a working simulation that demonstrates how autonomous vehicles interact with infrastructure safely.

This project highlighted the power of UPPAAL-SMC in modeling and analyzing **real-world autonomous systems**, providing both a **practical case study** and a foundation for further extensions (e.g., more cars, multiple intersections, emergency vehicles).