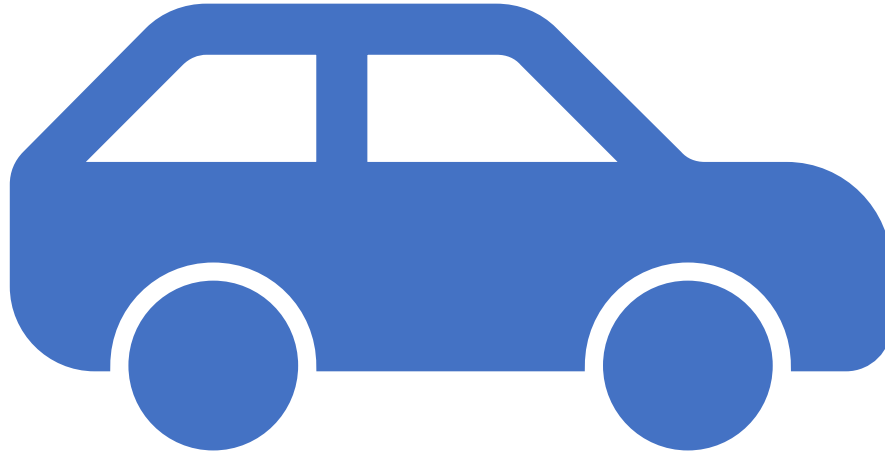



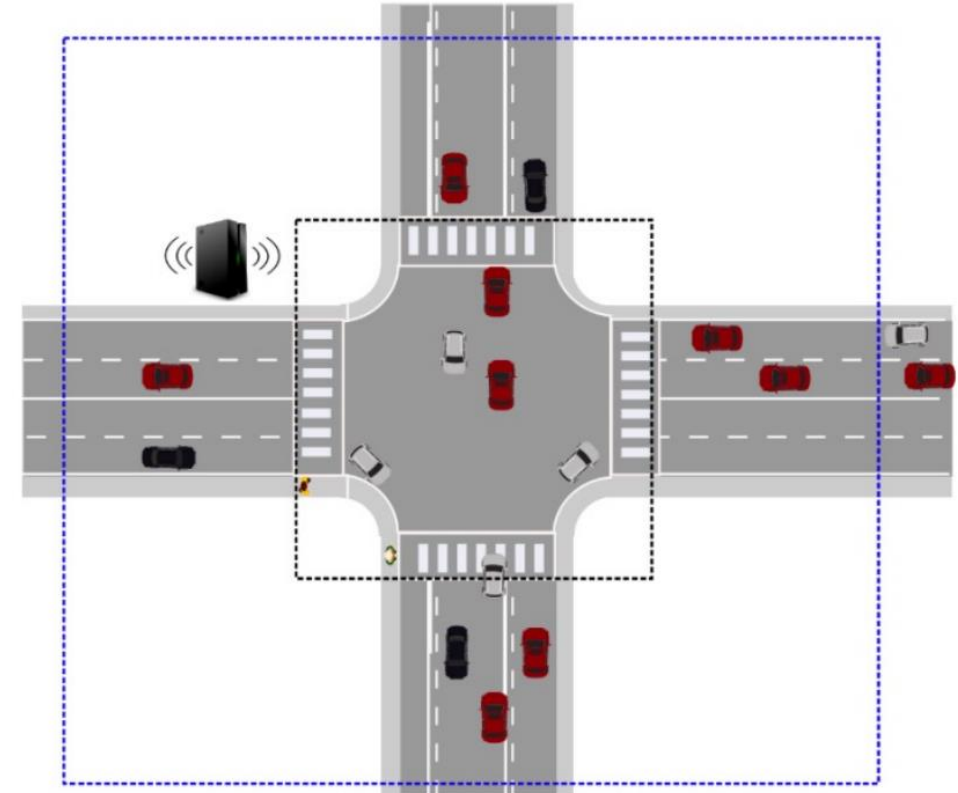
Optimised Traffic Management

- Moiz Zaheer Malik
- Jonathan Bahry
- Rubayet Kamal
- Mofifoluwa Akinwande

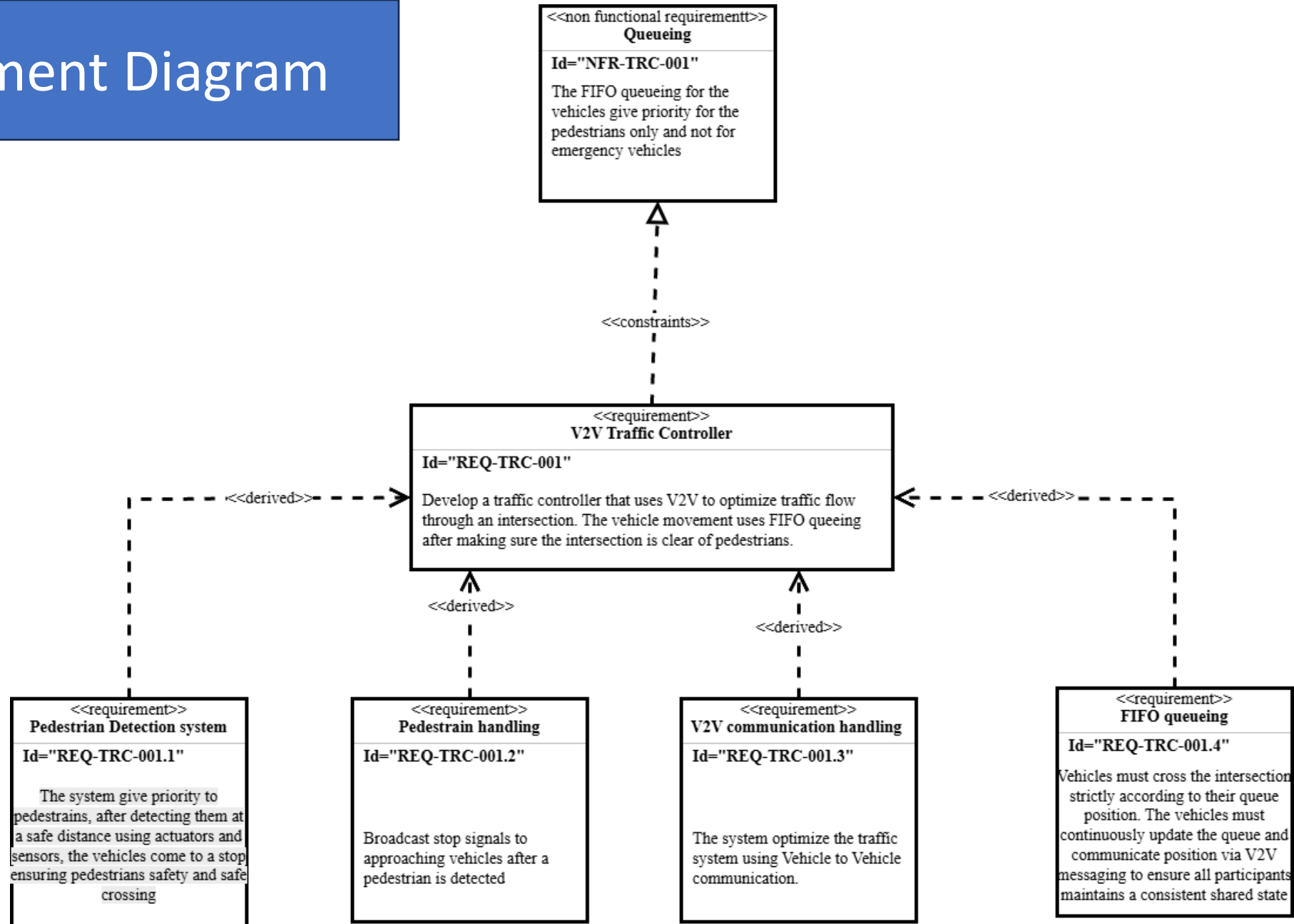


Motivation

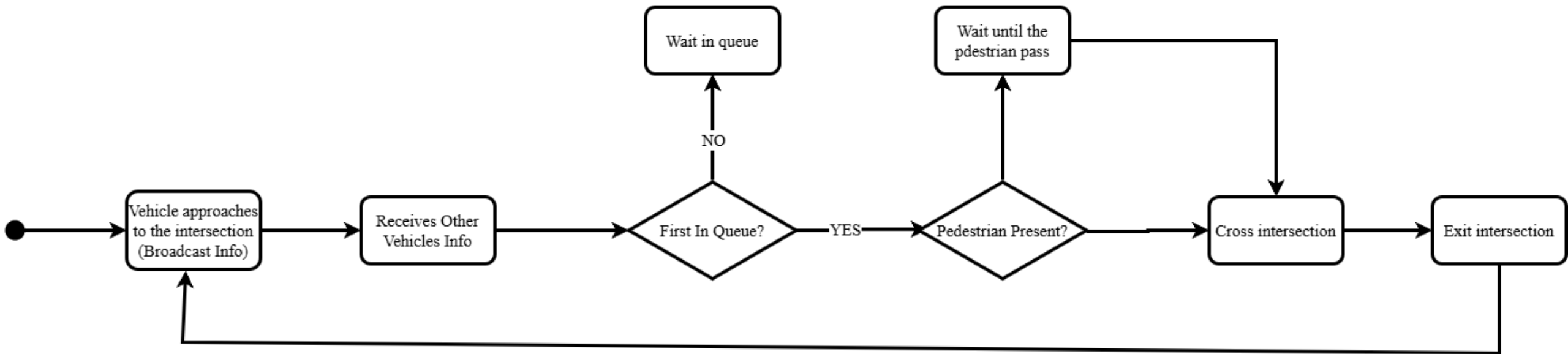
- Efficient Cross-Traffic Simulation for Autonomous Cars:
 - Cars to stop at intersection for pedestrians.
 - Cars to use queue system to avoid collision with other cars.
 - Implementation with FreeRTOS.
 - Hardware realisation with VHDL.
- 



Requirement Diagram

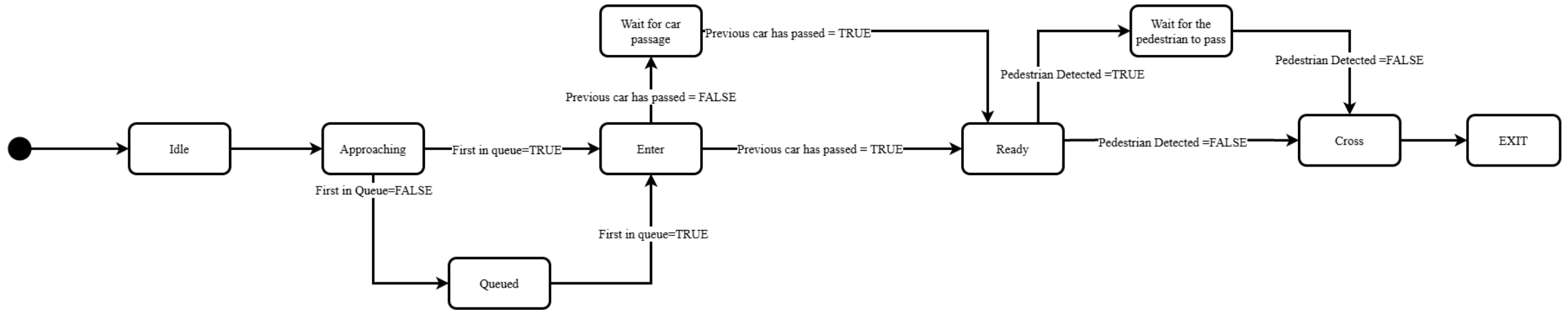


Activity Diagram for Normal Mode

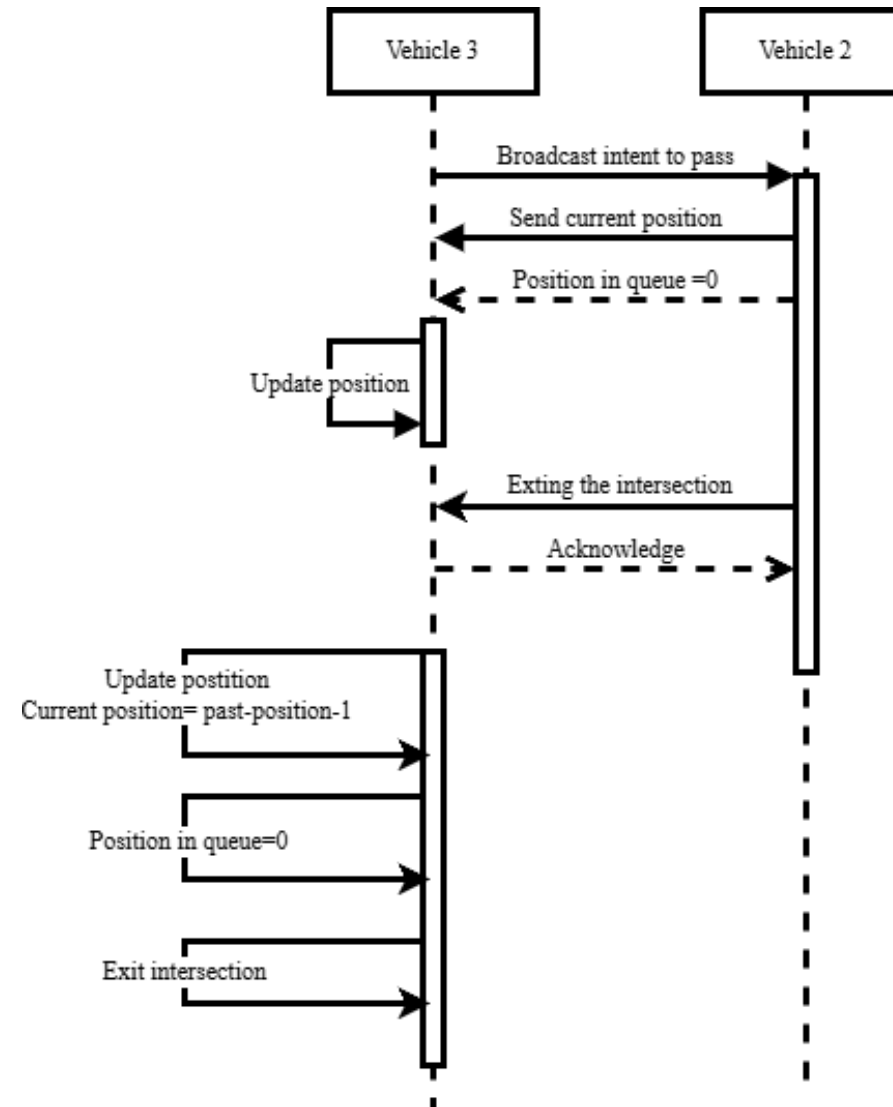


State Machine Diagram

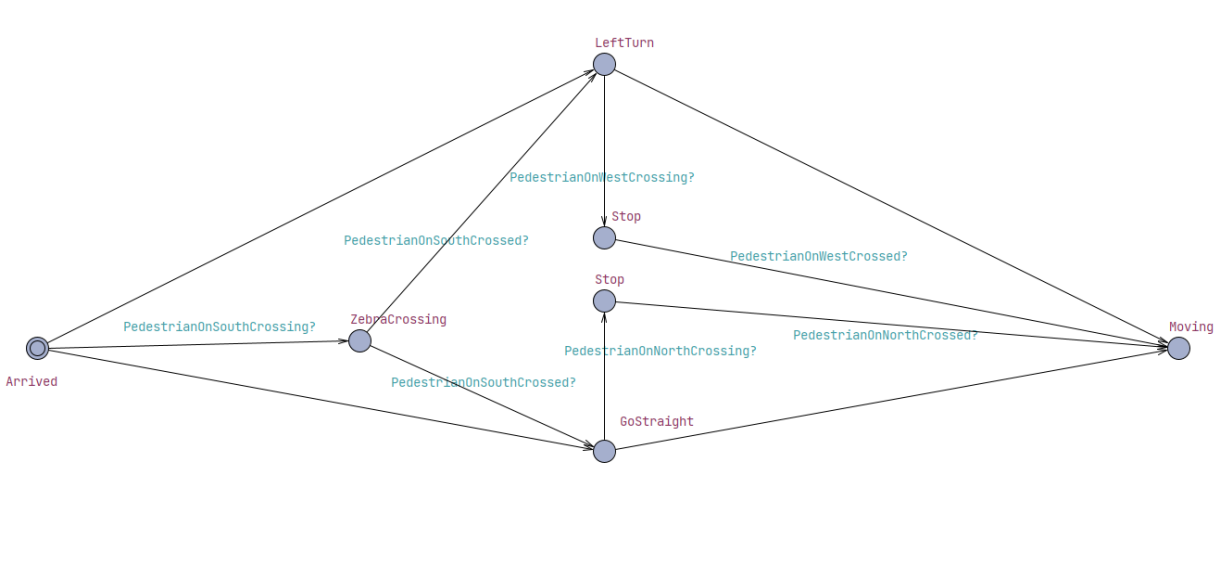
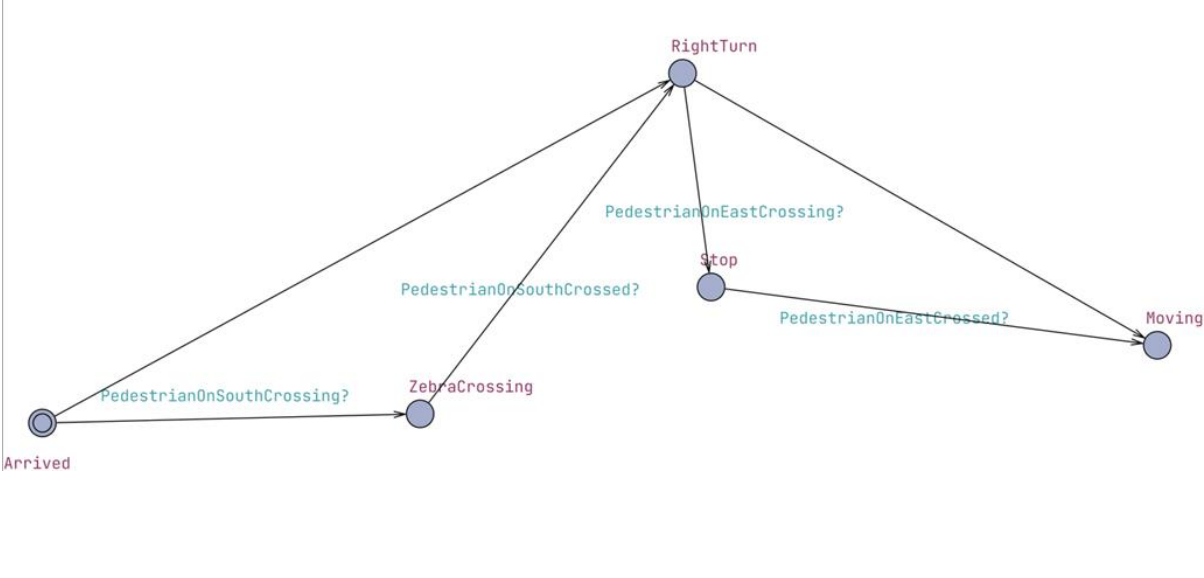
State machine diagram



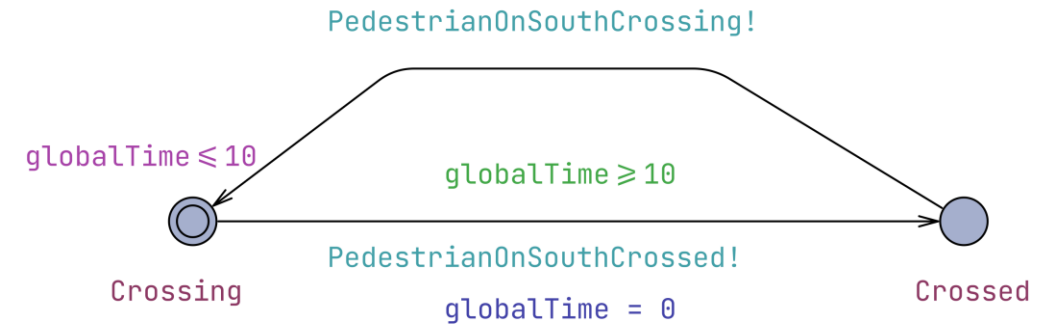
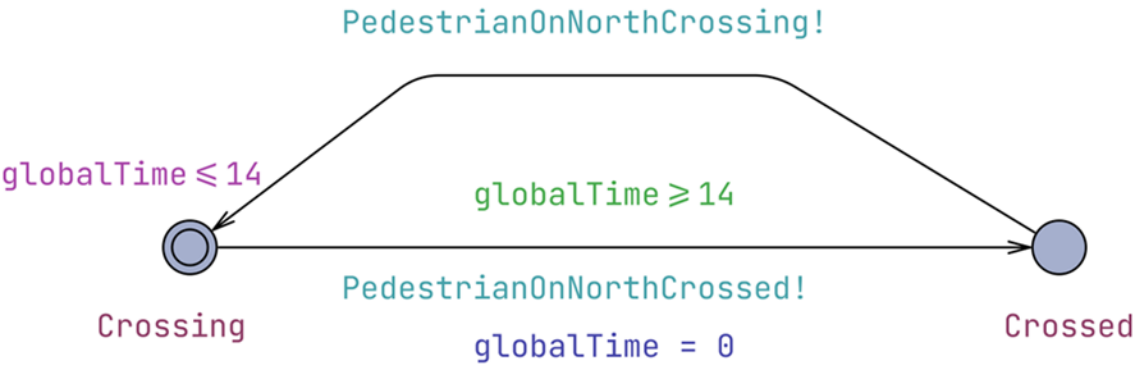
Sequence diagrams for Queue communication



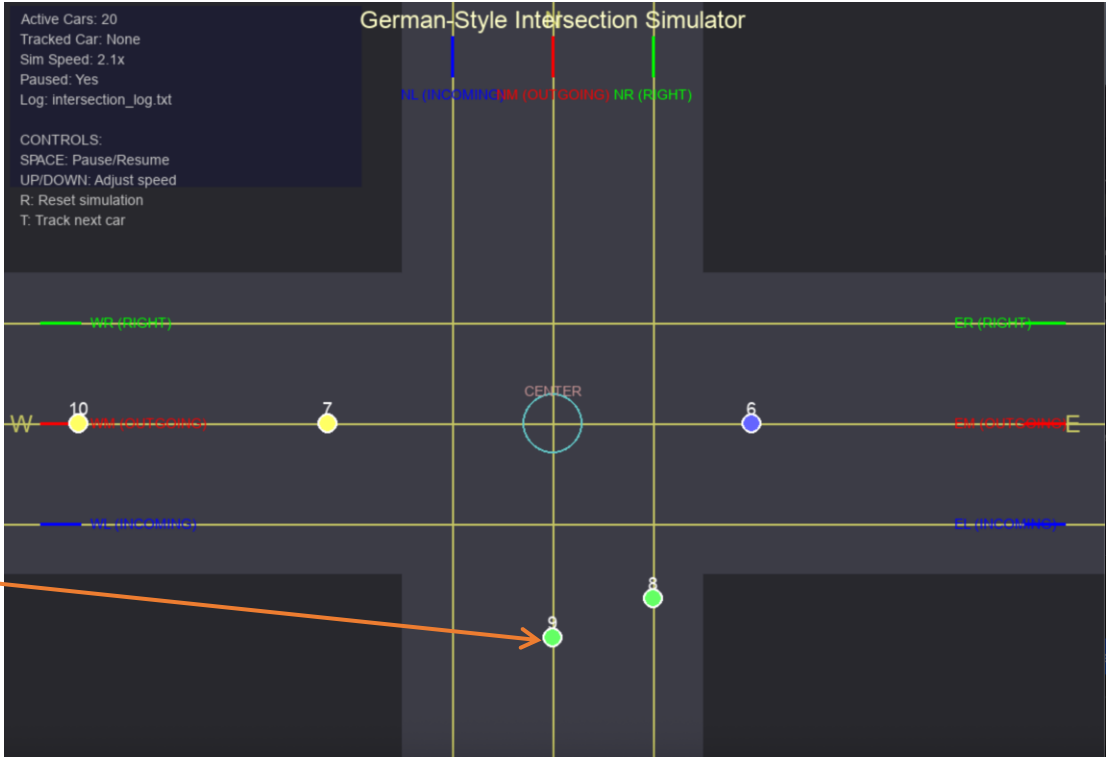
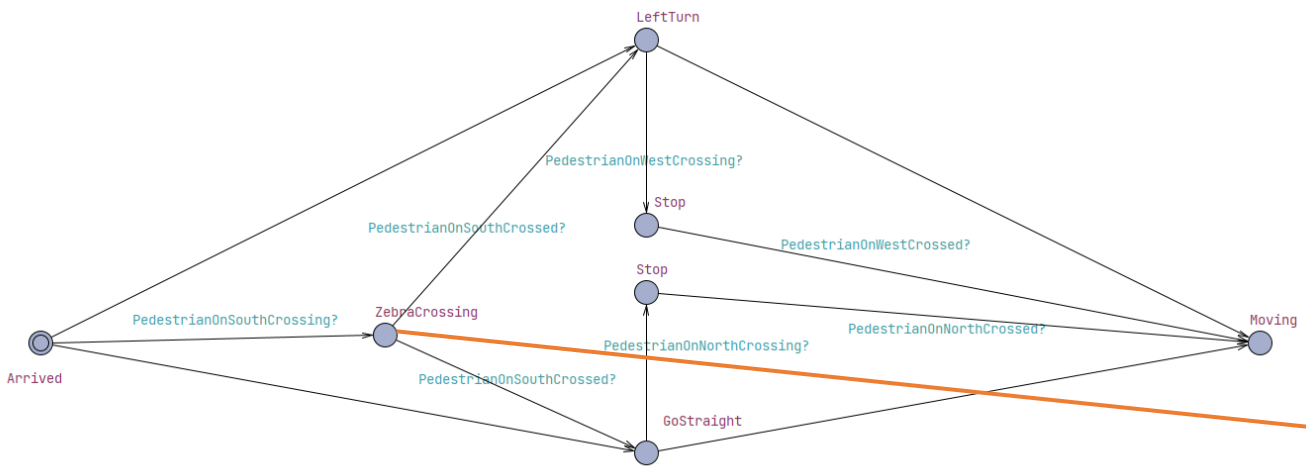
UPPAAL Models for Car Movement in South Lane



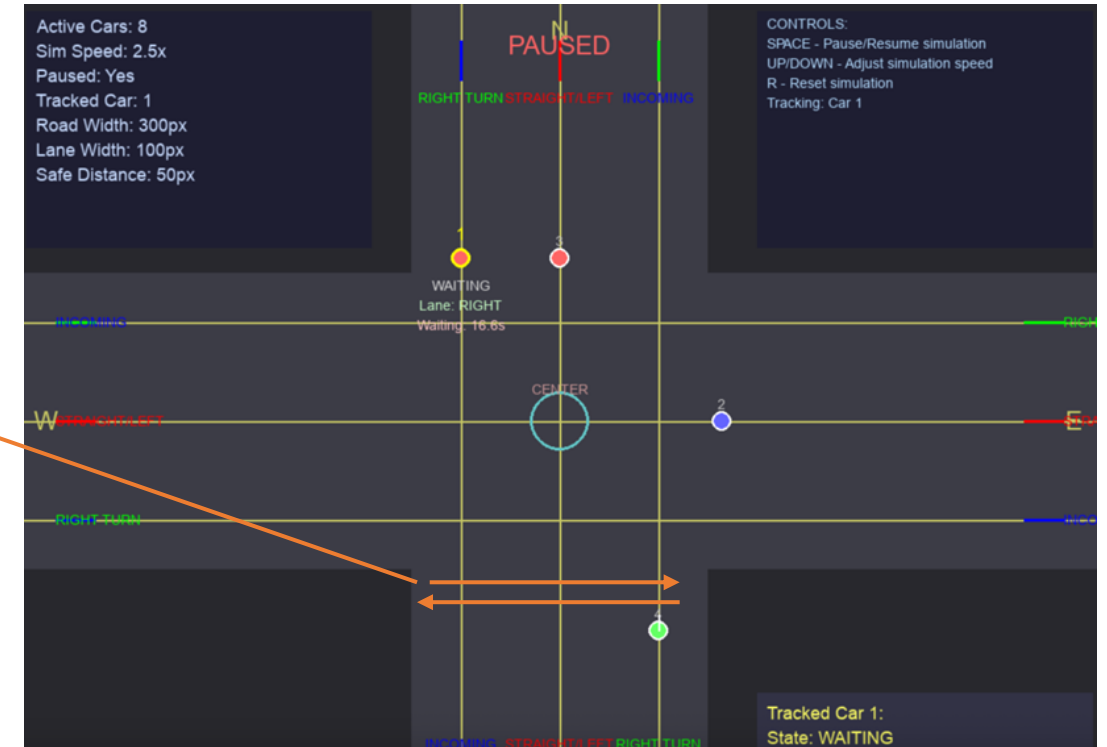
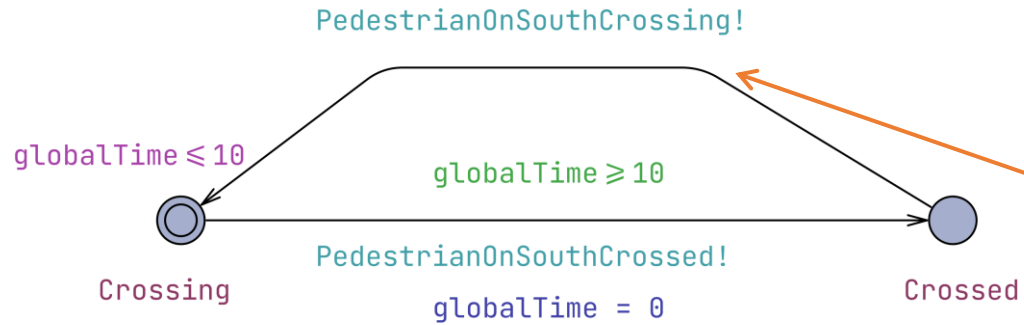
UPPAAL Models for Pedestrian Movement



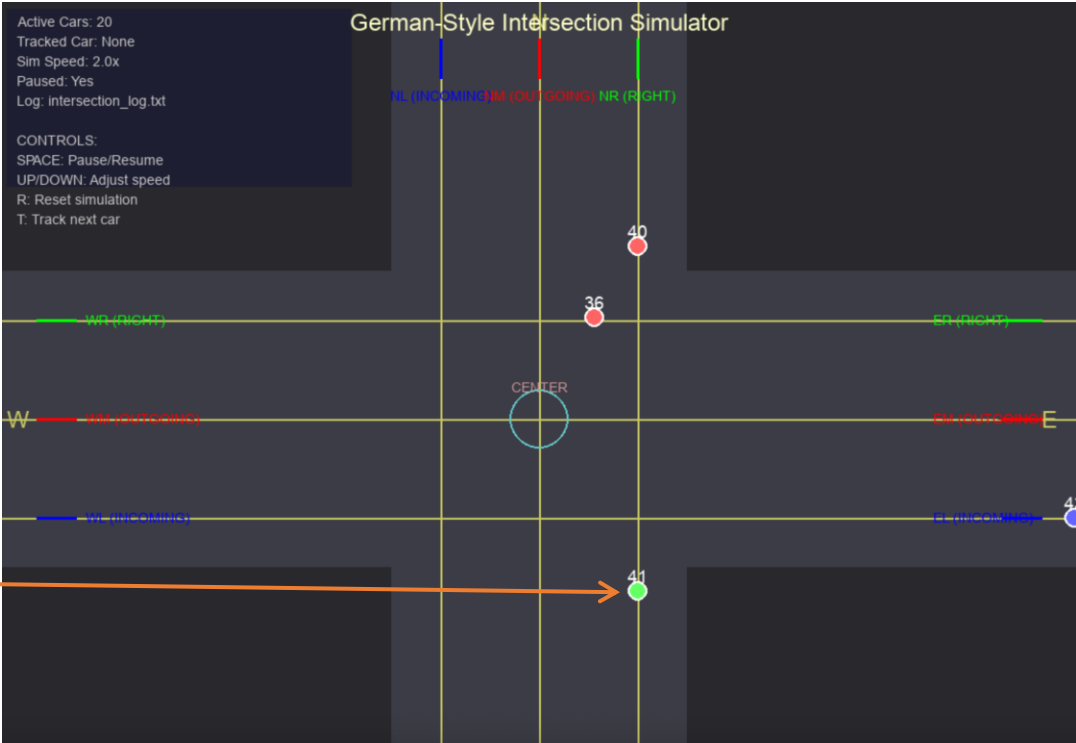
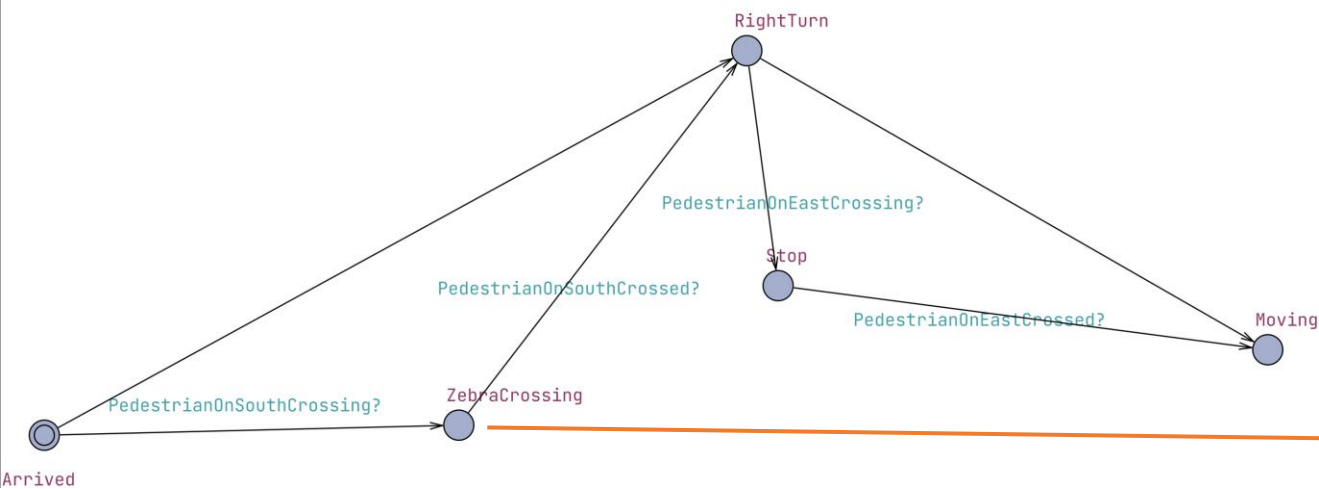
UPPAAL Model for Car in middle-lane of South



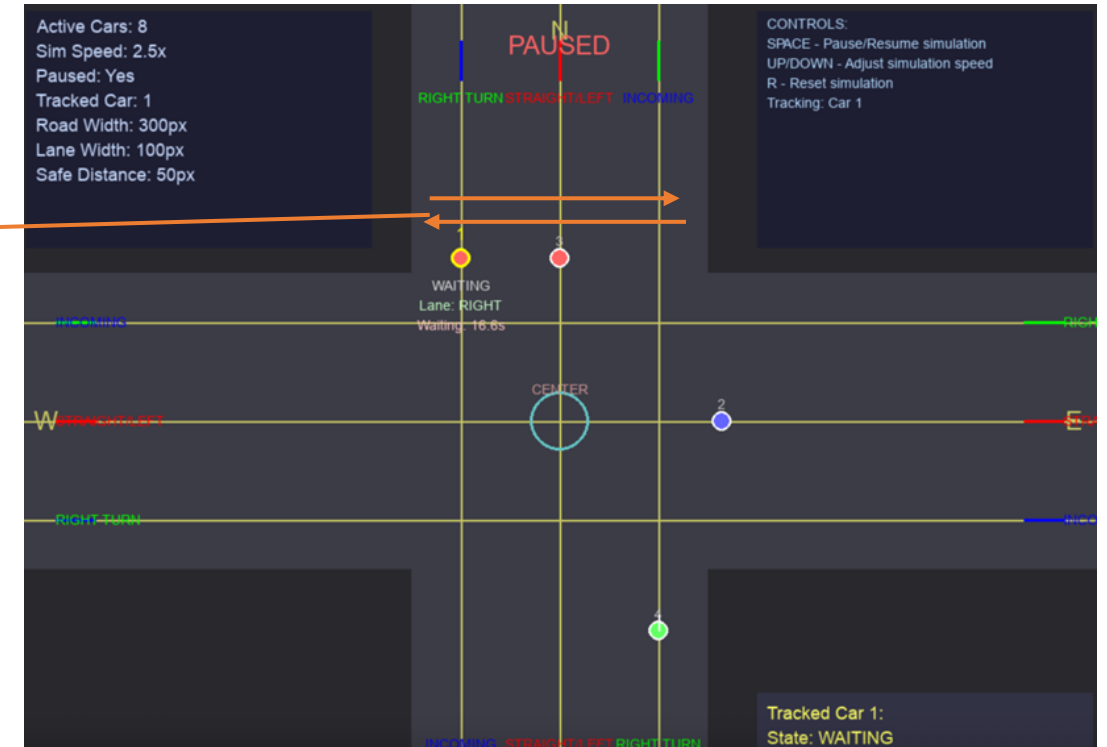
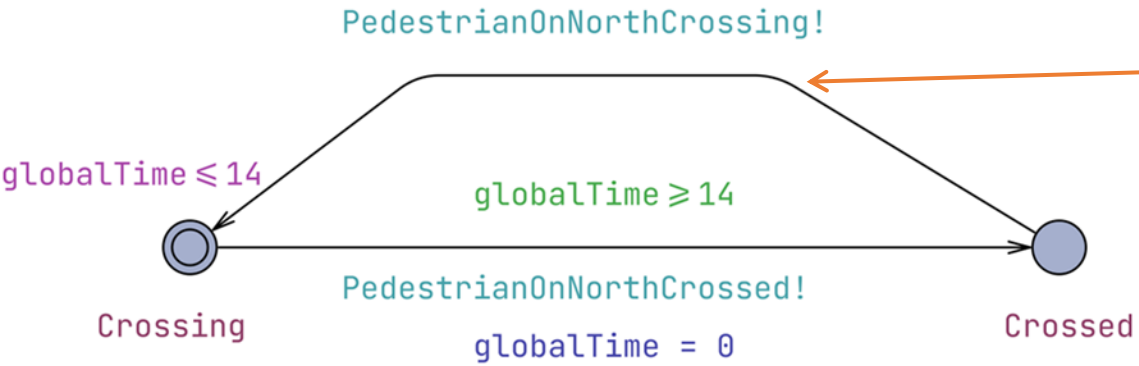
UPPAAL Model for Pedestrian crossing On South



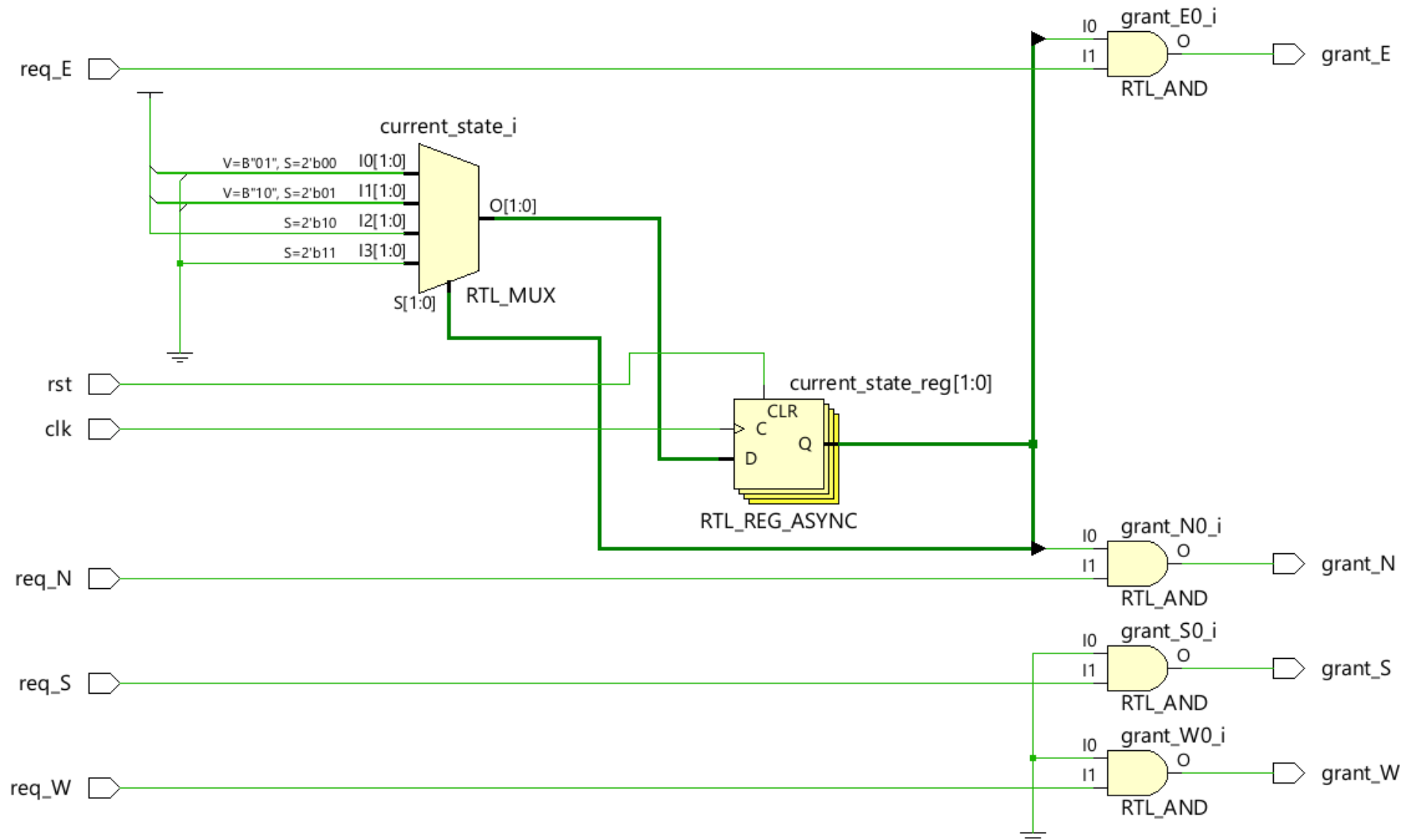
UPPAAL Model for Car in right-lane of South



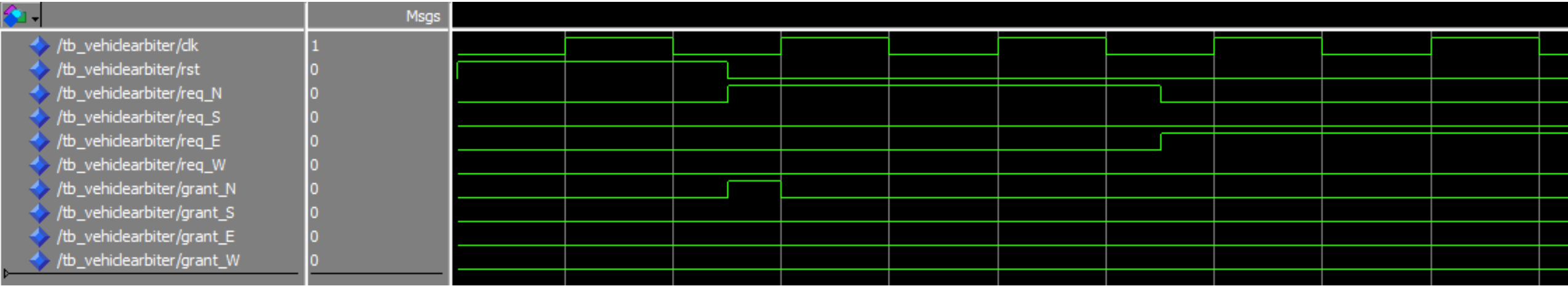
UPPAAL Model for Pedestrian crossing On North



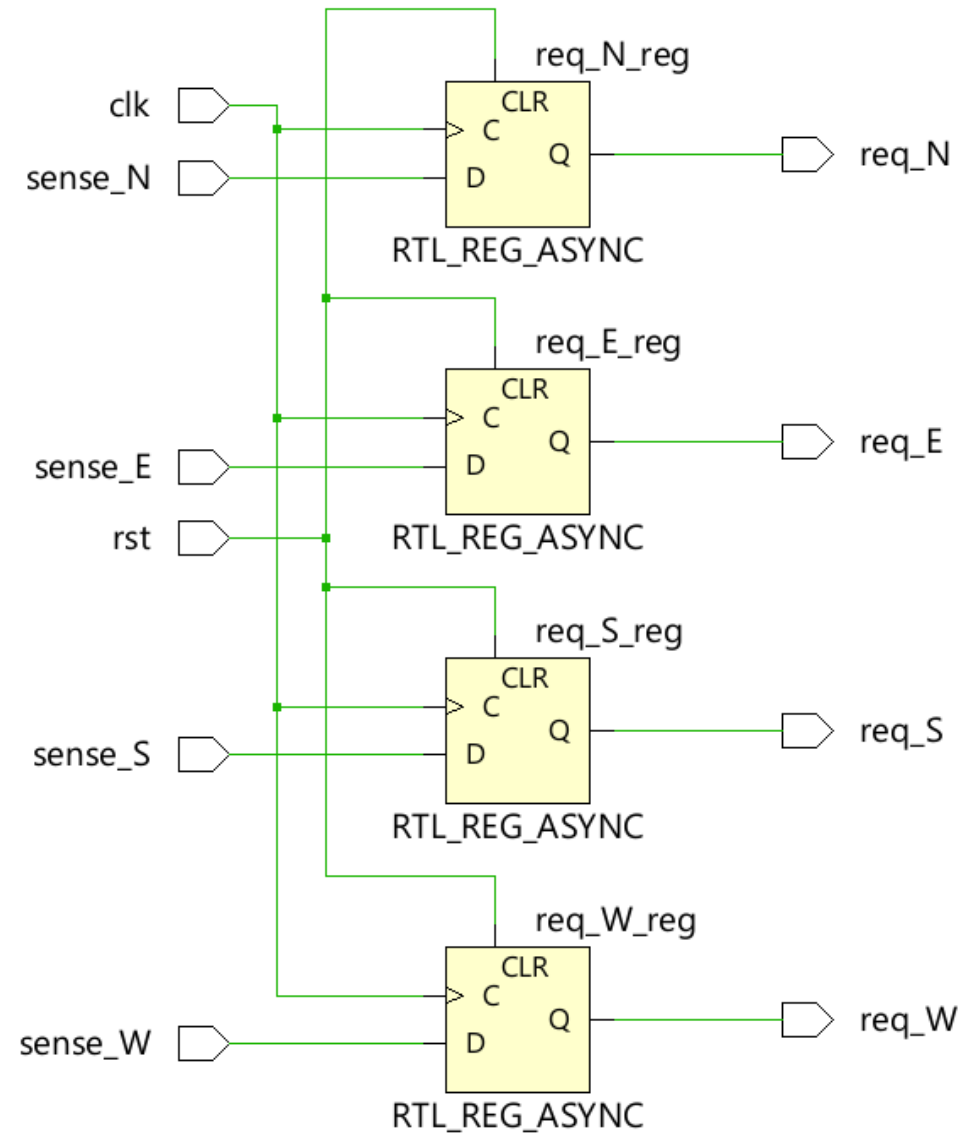
Vehicle Arbiter Module



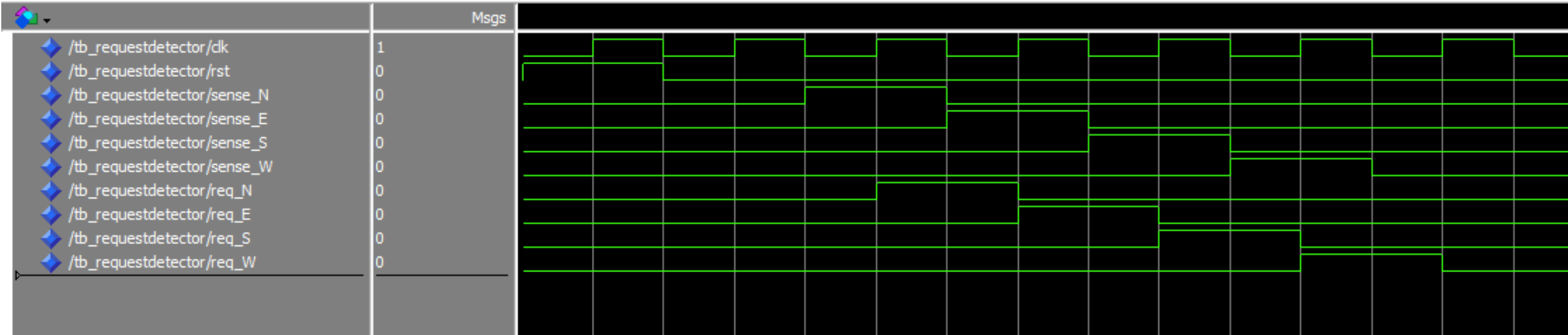
Hardware Implementation: Vehicle Arbiter



Request Detection Logic

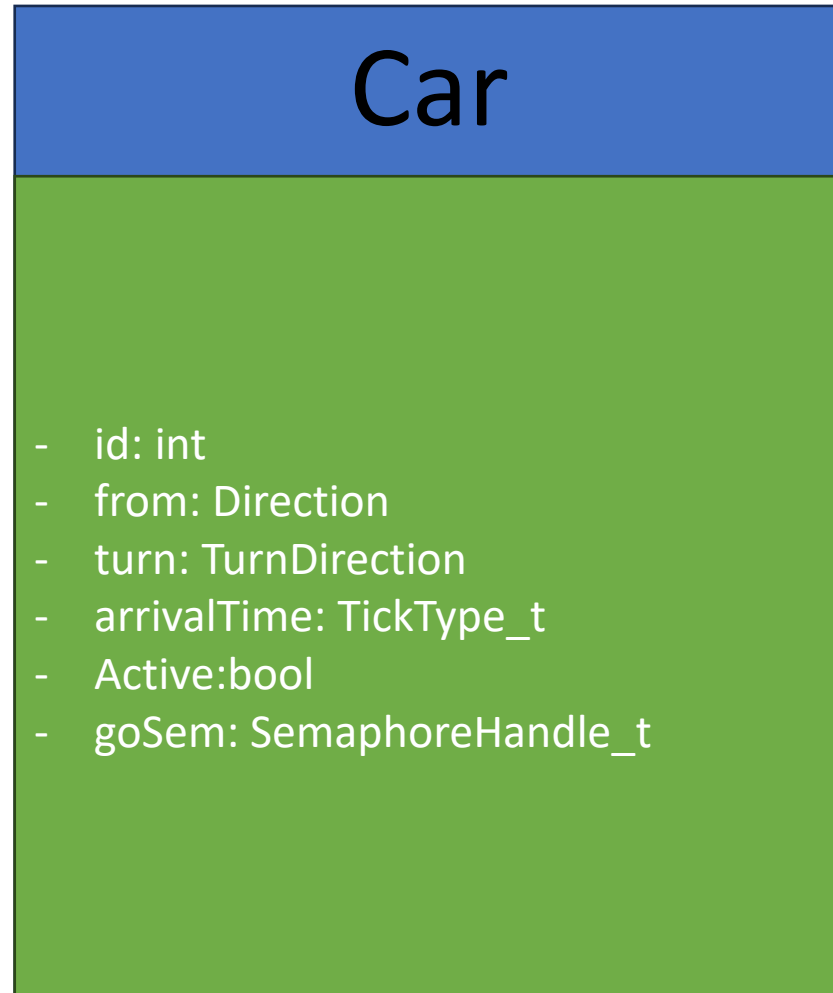


Hardware Implementation: Request Detector

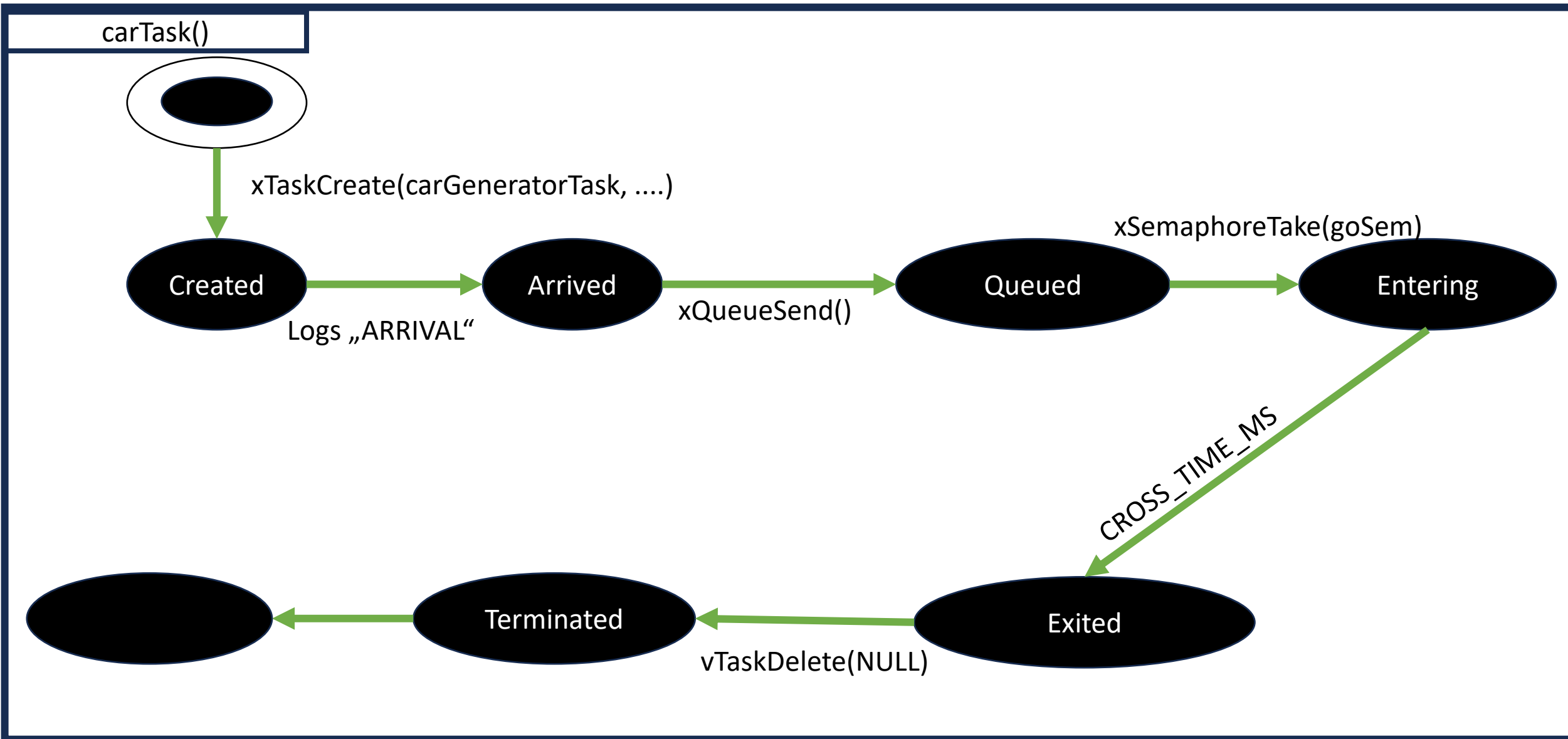


Car Parameters

ClassDiagram

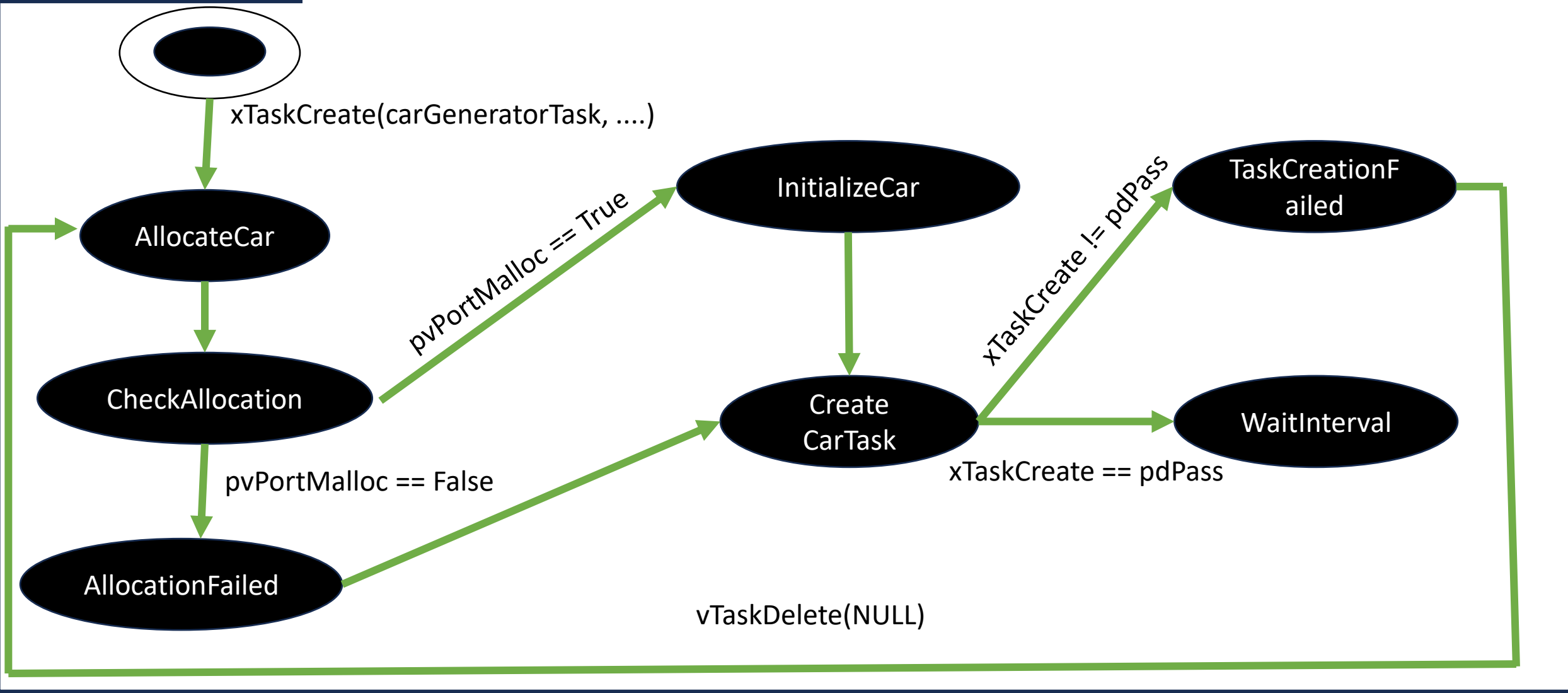


Software Implementation: FreeRTOS



Software Implementation: FreeRTOS

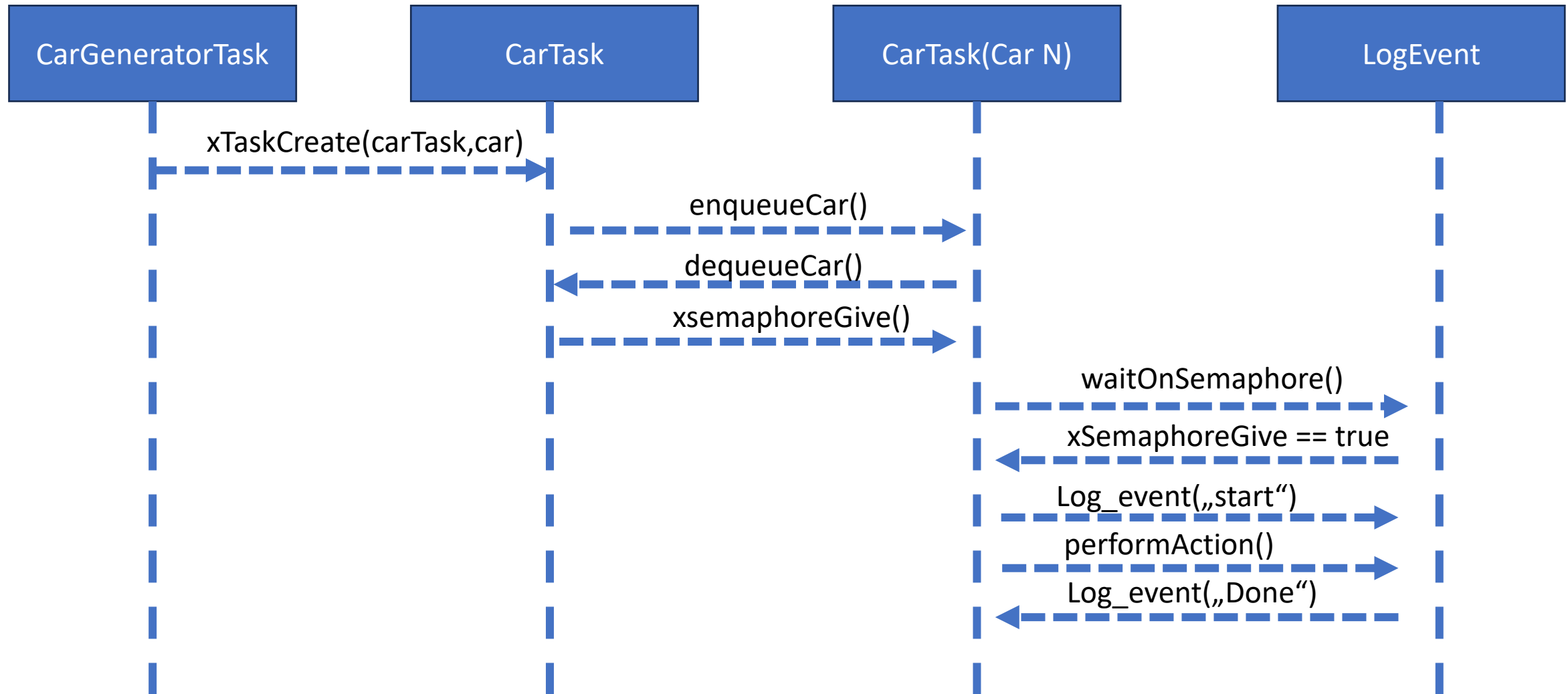
carGeneratorTask()



Conflicts at Intersection?

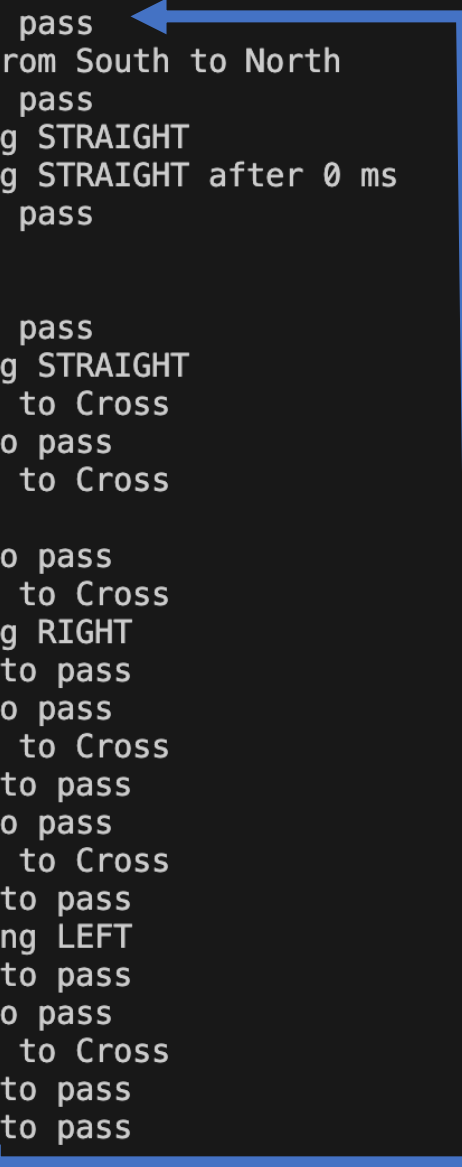
- Resolved using First come First serve (FCFS) technique.
- Arbiter Task responsible for:
 - listening to all head of all queues
 - deciding which has the earliest arrival time
 - releasing queues for movement.
- Cars can only cross when they receive the go from the Arbiter.

Implementation Algorithm: FCFS



Challenges:

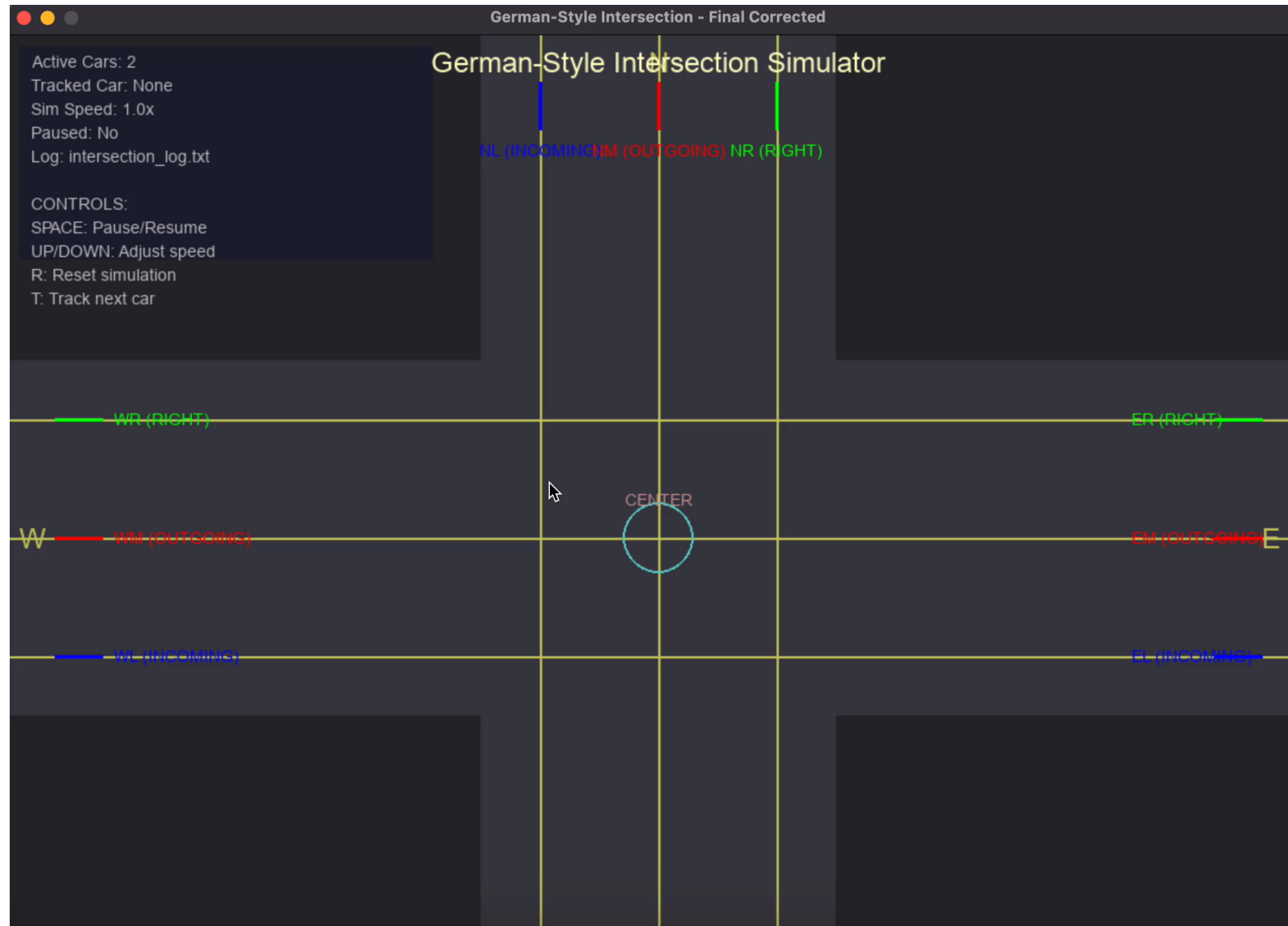
```
Car 7 is waiting for Car 6 to pass
Pedestrian On West Crossing from South to North
Car 7 is waiting for Car 6 to pass
Car 9 arrived from NORTH going STRAIGHT
Car 9 ENTERED from NORTH going STRAIGHT after 0 ms
Car 7 is waiting for Car 6 to pass
Car 6 EXITED
Car 8 EXITED
Car 7 is waiting for Car 9 to pass
Car 10 arrived from WEST going STRAIGHT
Car 10 waiting for Pedestrian to Cross
Car 7 is waiting for Car 10 to pass
Car 10 waiting for Pedestrian to Cross
Car 9 EXITED
Car 7 is waiting for Car 10 to pass
Car 10 waiting for Pedestrian to Cross
Car 11 arrived from EAST going RIGHT
Car 11 is waiting for Car 10 to pass
Car 7 is waiting for Car 10 to pass
Car 10 waiting for Pedestrian to Cross
Car 11 is waiting for Car 10 to pass
Car 7 is waiting for Car 10 to pass
Car 10 waiting for Pedestrian to Cross
Car 11 is waiting for Car 10 to pass
Car 12 arrived from NORTH going LEFT
Car 12 is waiting for Car 10 to pass
Car 7 is waiting for Car 10 to pass
Car 10 waiting for Pedestrian to Cross
Car 11 is waiting for Car 10 to pass
Car 12 is waiting for Car 10 to pass
Pedestrian crossing cleared
```



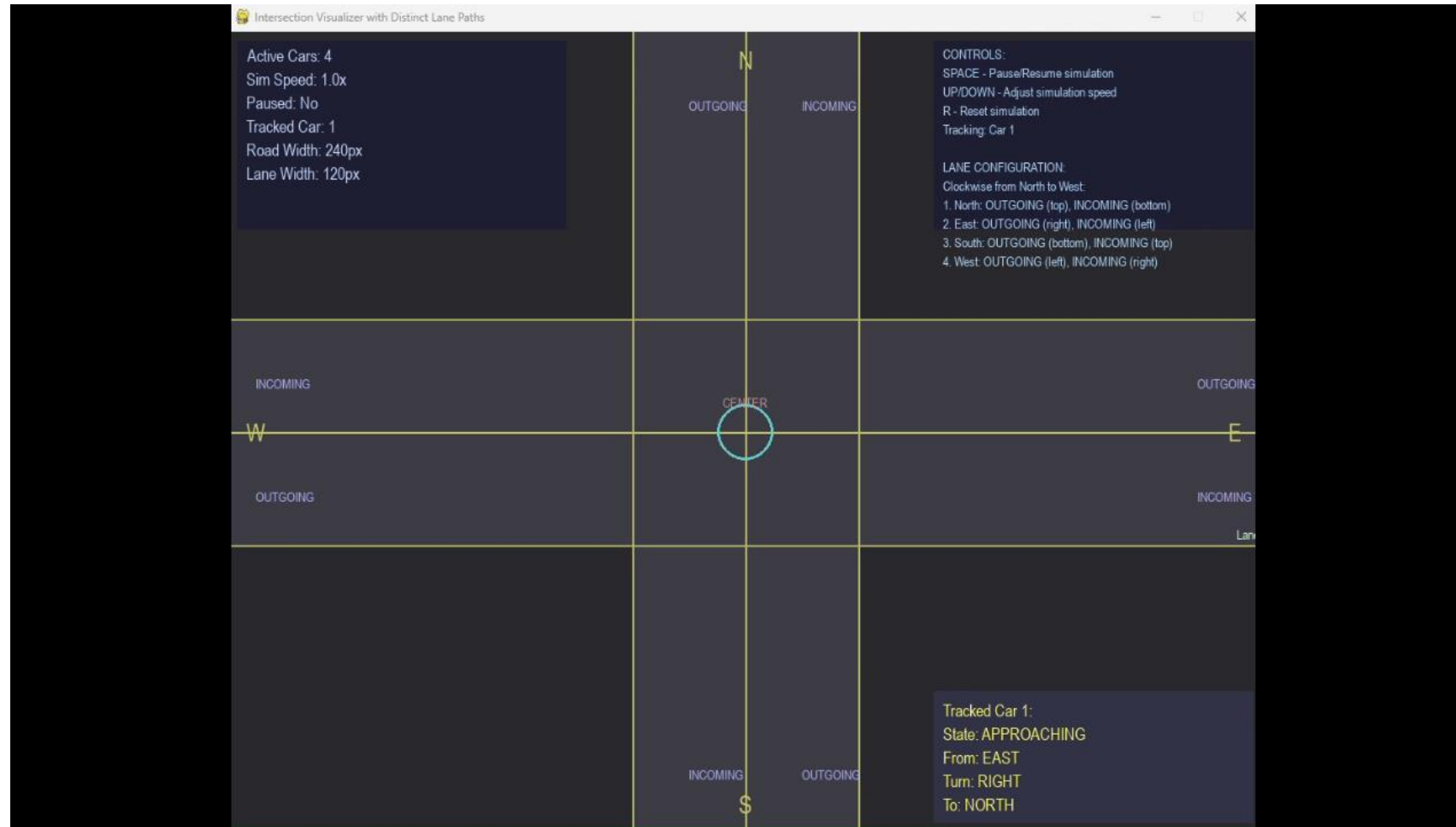
Challenges:

```
Car 1 arrived from WEST going STRAIGHT
Car 1 ENTERED from WEST going STRAIGHT after 0 ms
Car 2 arrived from SOUTH going STRAIGHT
Car 2 ENTERED from SOUTH going STRAIGHT after 0 ms
Car 1 EXITED
Car 3 arrived from NORTH going LEFT
Car 3 is waiting for Car 2 to pass
Car 3 is waiting for Car 2 to pass
Car 2 EXITED
Car 3 ENTERED from NORTH going LEFT after 1000 ms
Car 4 arrived from SOUTH going RIGHT
Car 4 ENTERED from SOUTH going RIGHT after 0 ms
Car 5 arrived from EAST going LEFT
Car 5 is waiting for Car 3 to pass
Car 5 is waiting for Car 3 to pass
Car 3 EXITED
Car 4 EXITED
```

Simulation (Unsuccessful Approach)



Simulation (Working Approach)



Summary and Future Work

- Implemented a four lane cross simulation using **queues**.
- **Software (FreeRTOS):** Used an Arbiter Task employing FCFS to resolve conflicts at Intersection.
- **Hardware (VHDL):** Designed request detection logic and vehicle arbiter module on hardware.
- **Future Work:**
 - Pedestrians detection.
 - Scaling lanes.