

# Autonomous Intersection Controller

- By
- Nirojan Navaratanarajah
  - Jaouaher Belgacem
  - Neaz Mahmud

# MOTIVATION

---

# Problematic

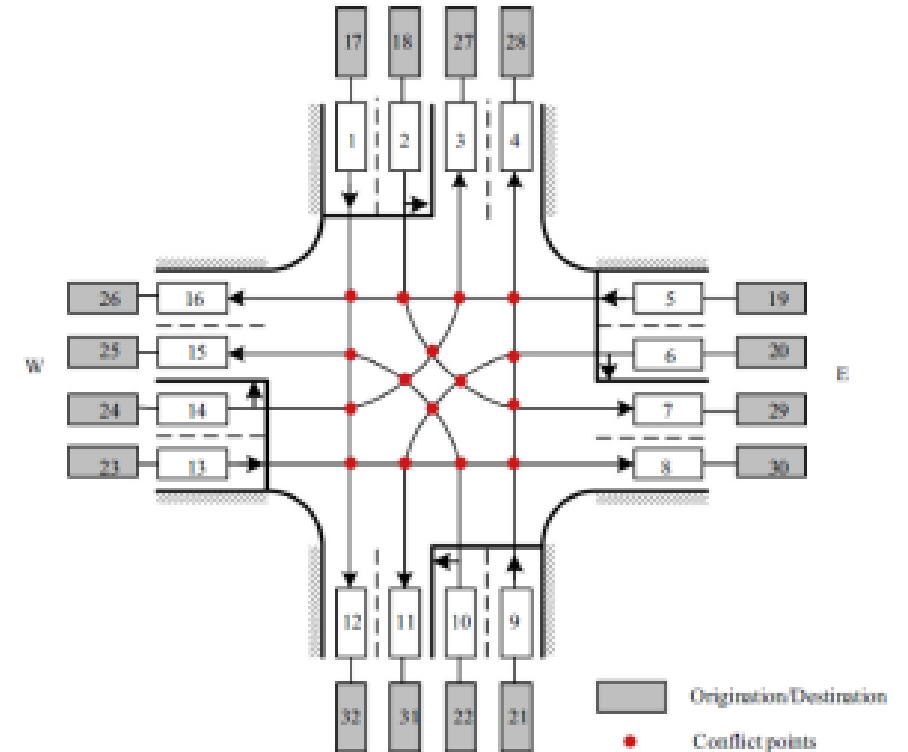
- **Entry/Destination possibilities**

*Entry North:* ➤ South  
➤ East  
➤ West

*Entry South:* ➤ North  
➤ East  
➤ West

*Entry West:* ➤ South  
➤ North  
➤ East

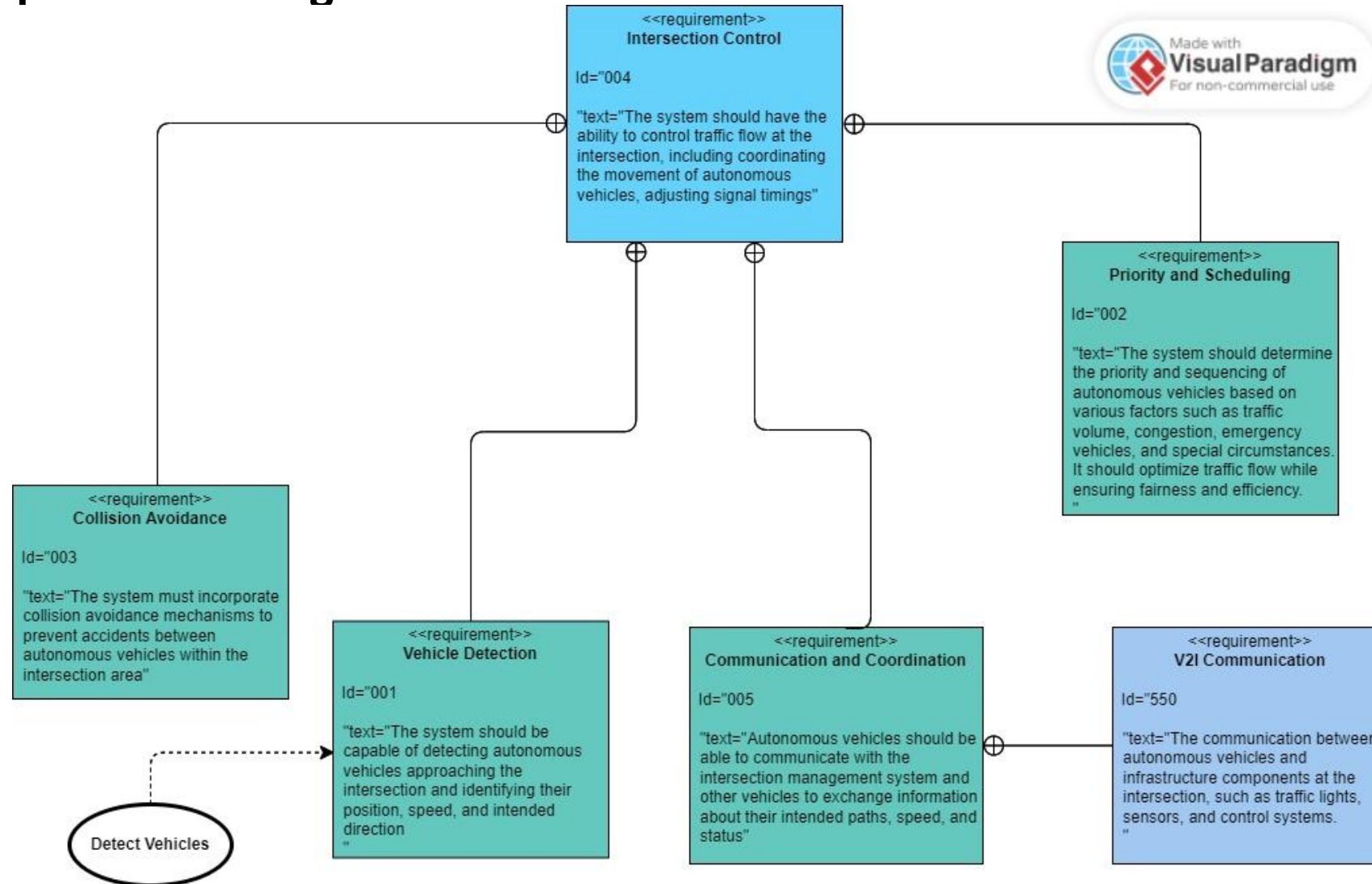
*Entry East:* ➤ South  
➤ North  
➤ West



[1]

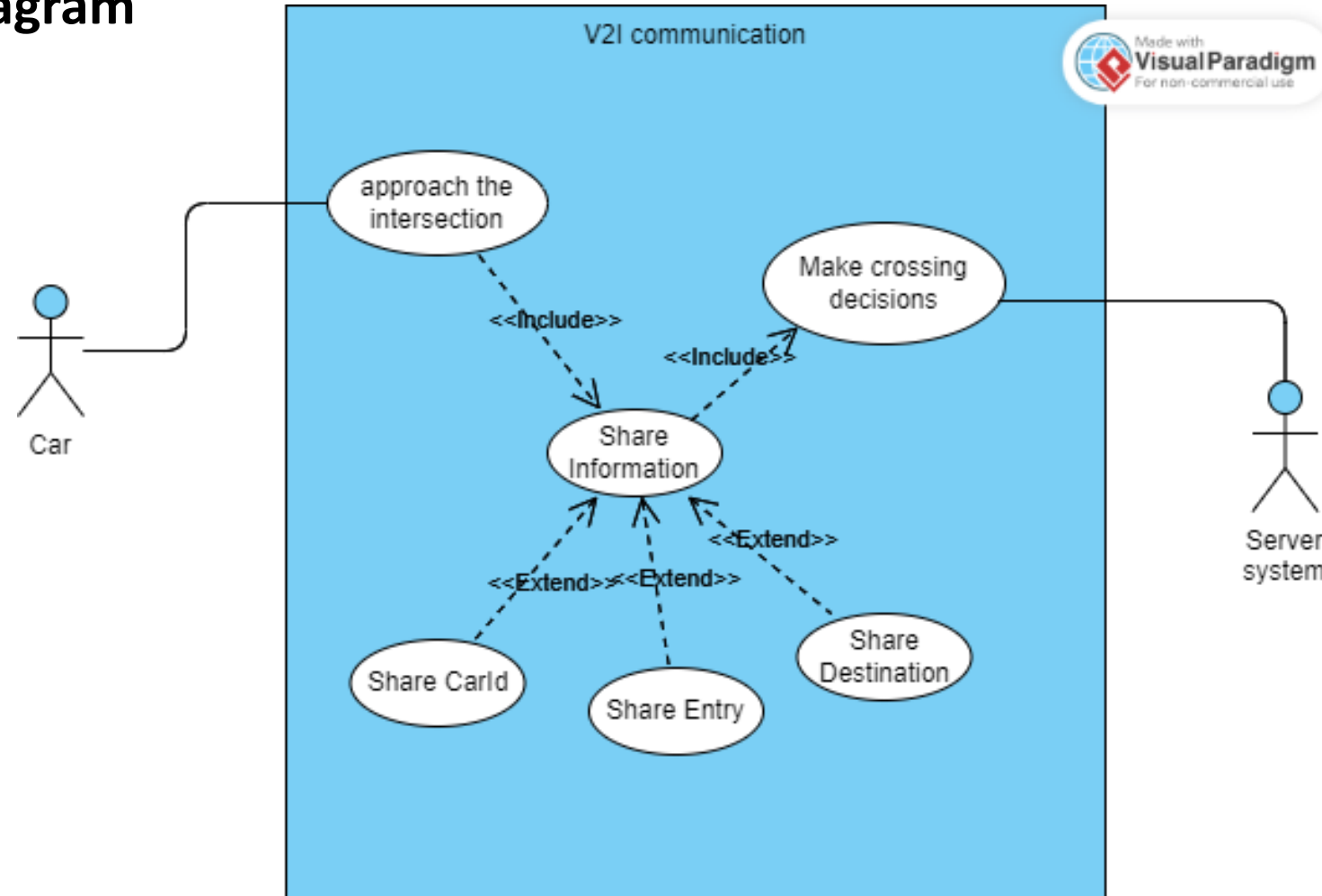
# Software Modelling

- Requirement Diagram



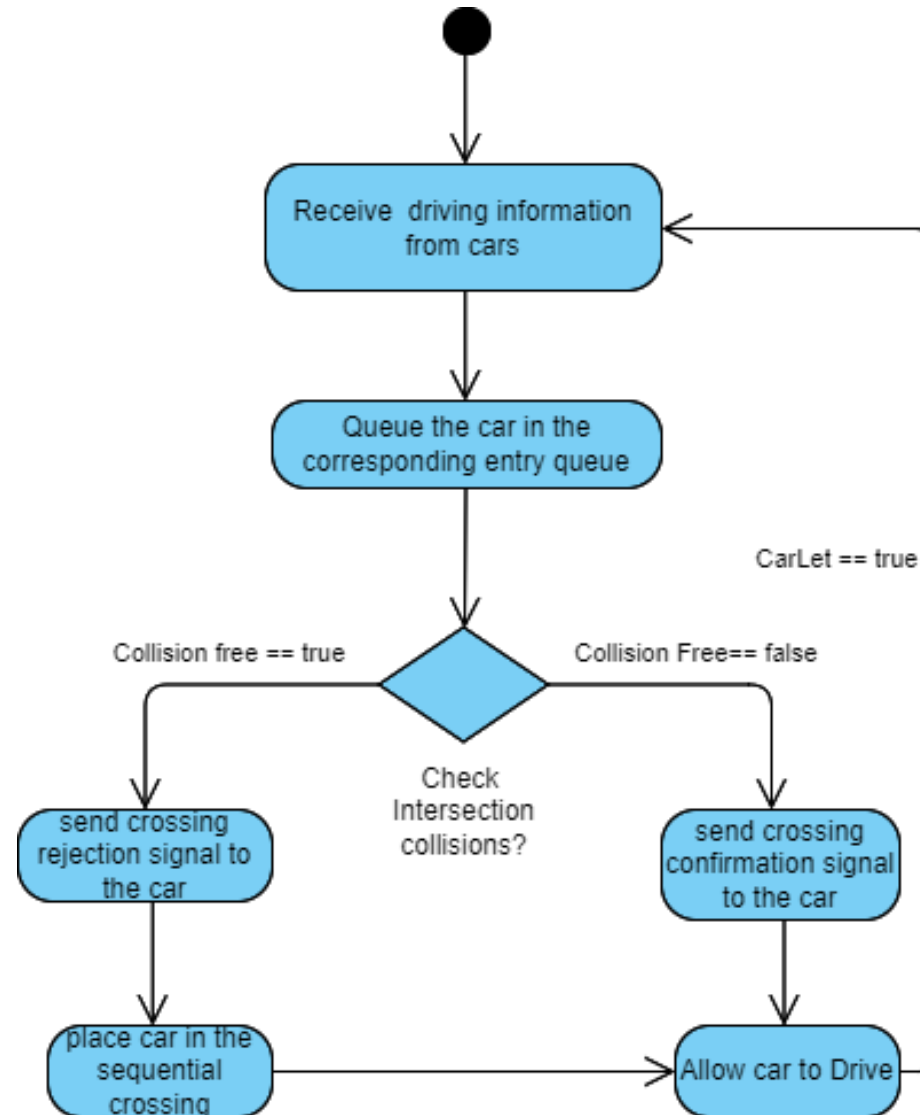
# Software Modelling

- Use Case Diagram



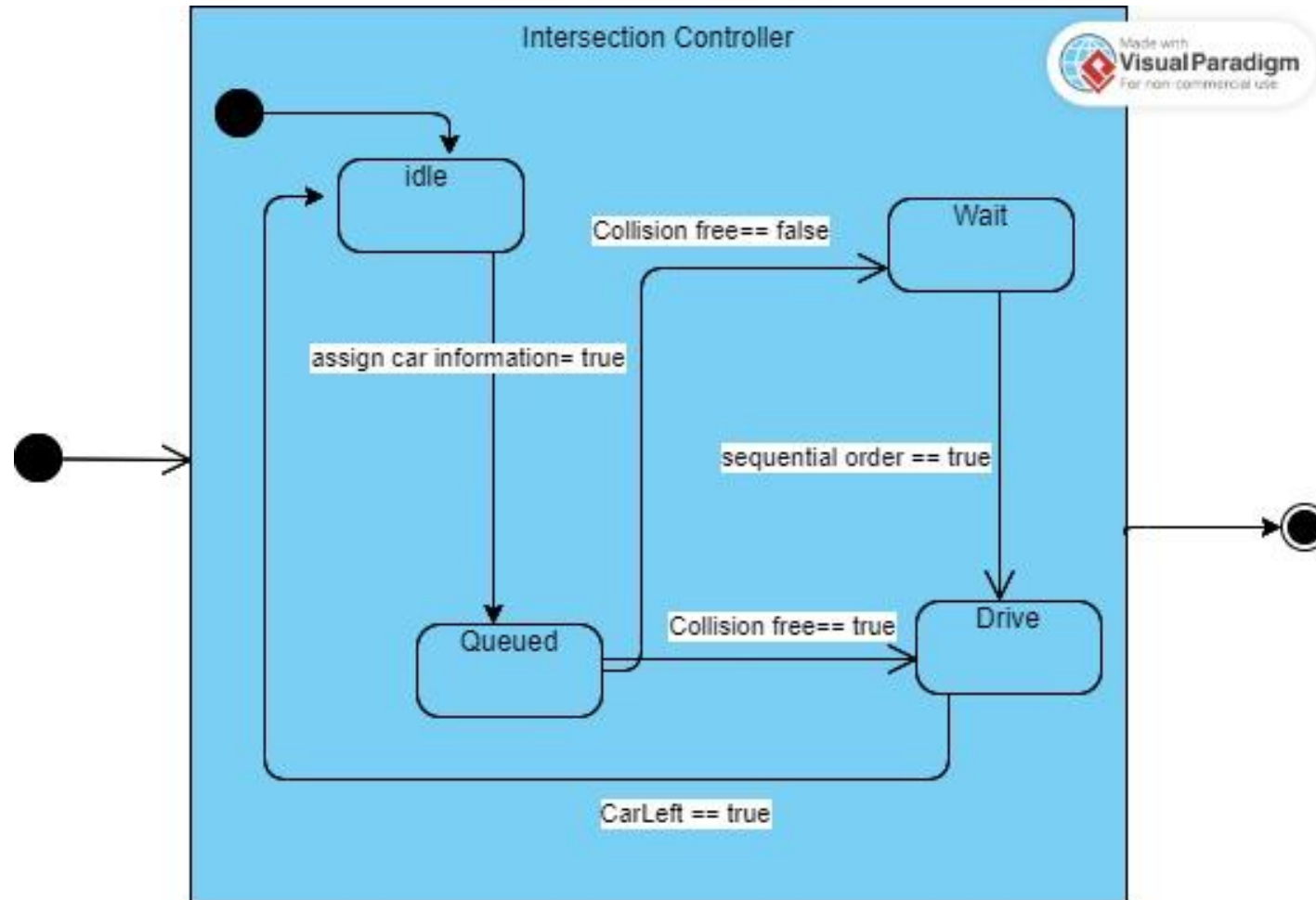
# Software Modelling

- Activity Diagram



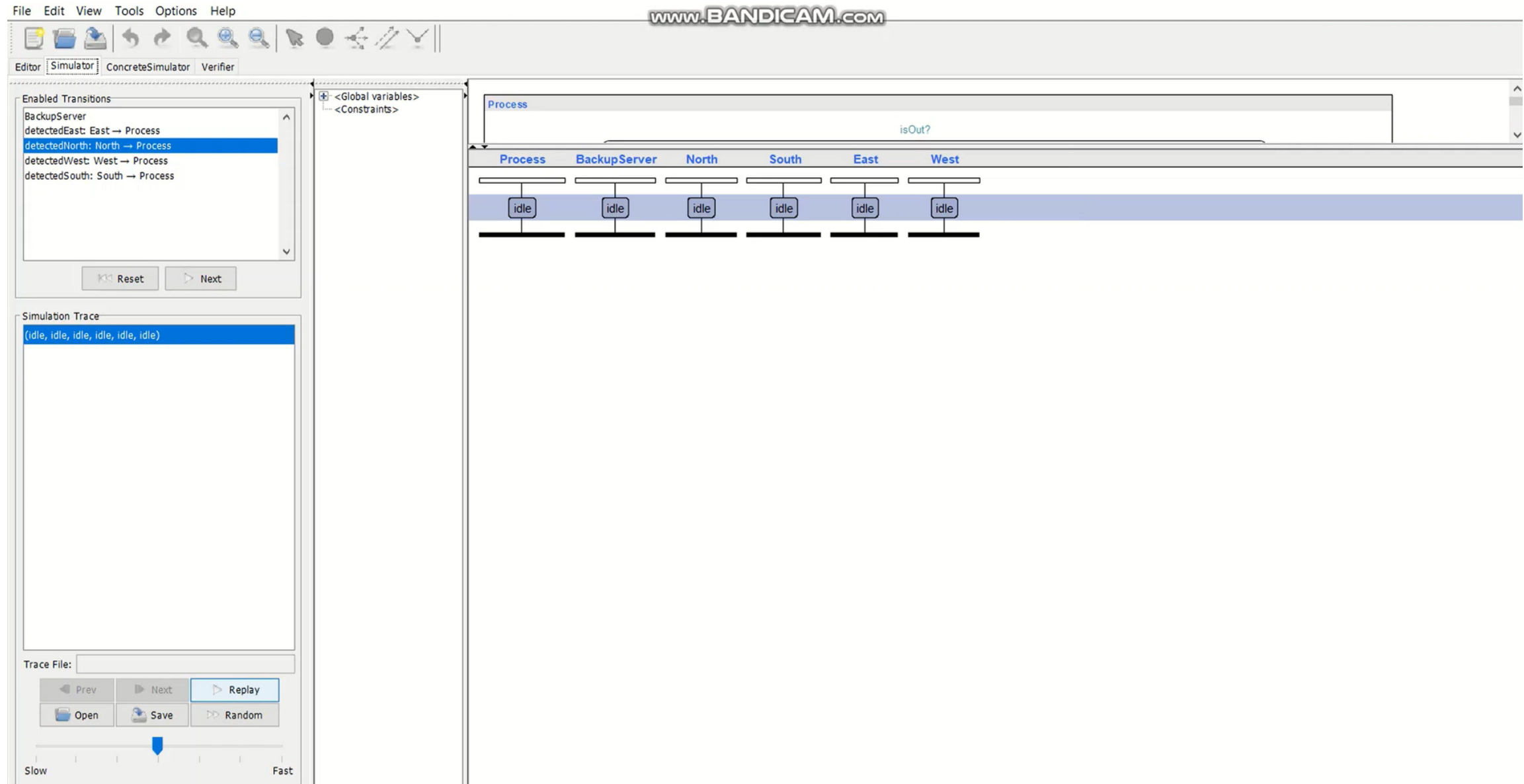
# Software Modelling

- State Machine Diagram



# Upaal Implementation

- System's simulation





# Uppaal Implementation

- System's verification

File Edit View Tools Options Help

www.BANDICAM.COM

Editor Simulator ConcreteSimulator Verifier

Overview

```
A[]!deadlock  
E<>North.out
```

Check  
Get Trace  
Insert Above  
Insert Below  
Remove  
Comments

Query

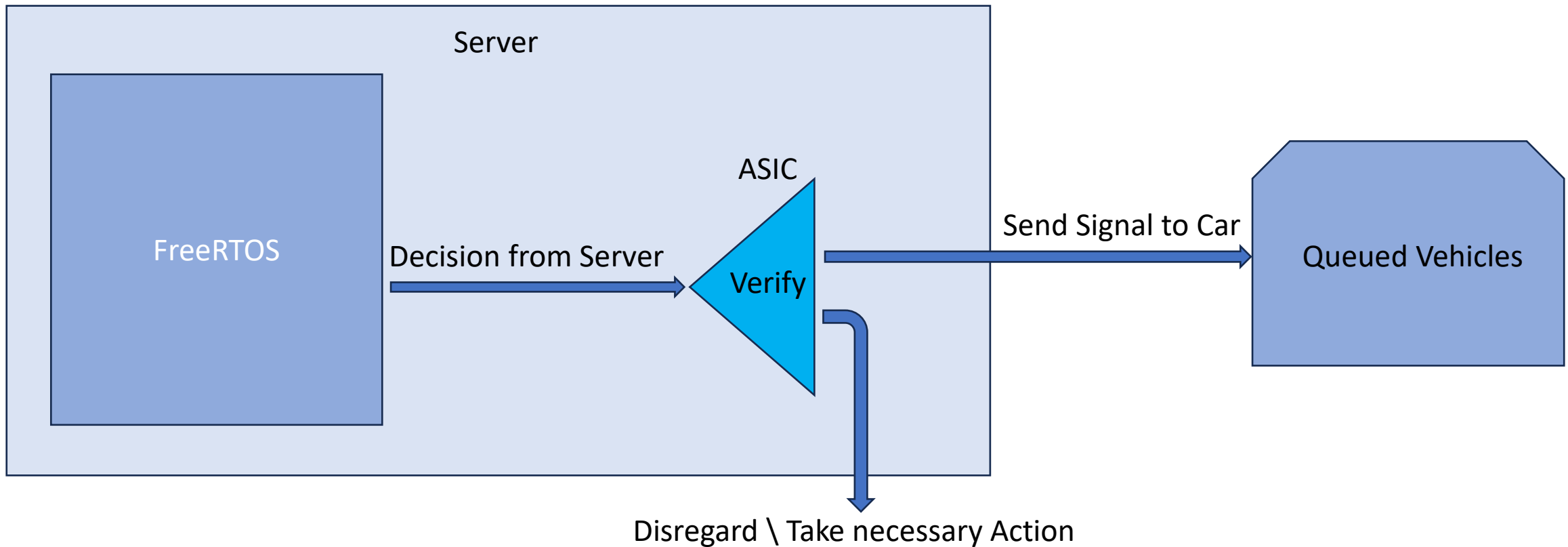
Comment

Status

(Academic) UPPAAL version 4.1.26-1 (rev. 7BCF30B7363A9518), February 2022 -- server.  
Disconnected.  
Established direct connection to local server.  
(Academic) UPPAAL version 4.1.26-1 (rev. 7BCF30B7363A9518), February 2022 -- server.

# Hw/Sw Codesign Implementation

- ASIC design



# Hw/Sw Codesign Implementation

- VHDL Code snippets

```
-- Scenario: Only cars coming from opposite directions can go straight at the same time
elsif (Car1 = '1' and Car2 = '1' and Car3 = '0' and Car4 = '0') then
    Car1Permission <= "111";
    Car2Permission <= "111";
    Car3Permission <= "000";
    Car4Permission <= "000";
    Collision <= '0';
```

```
elsif (Car1 = '0' and Car2 = '1' and Car3 = '1' and Car4 = '0') then
    Car1Permission <= "001";
    Car2Permission <= "111";
    Car3Permission <= "111";
    Car4Permission <= "111";
    Collision <= '1';
```

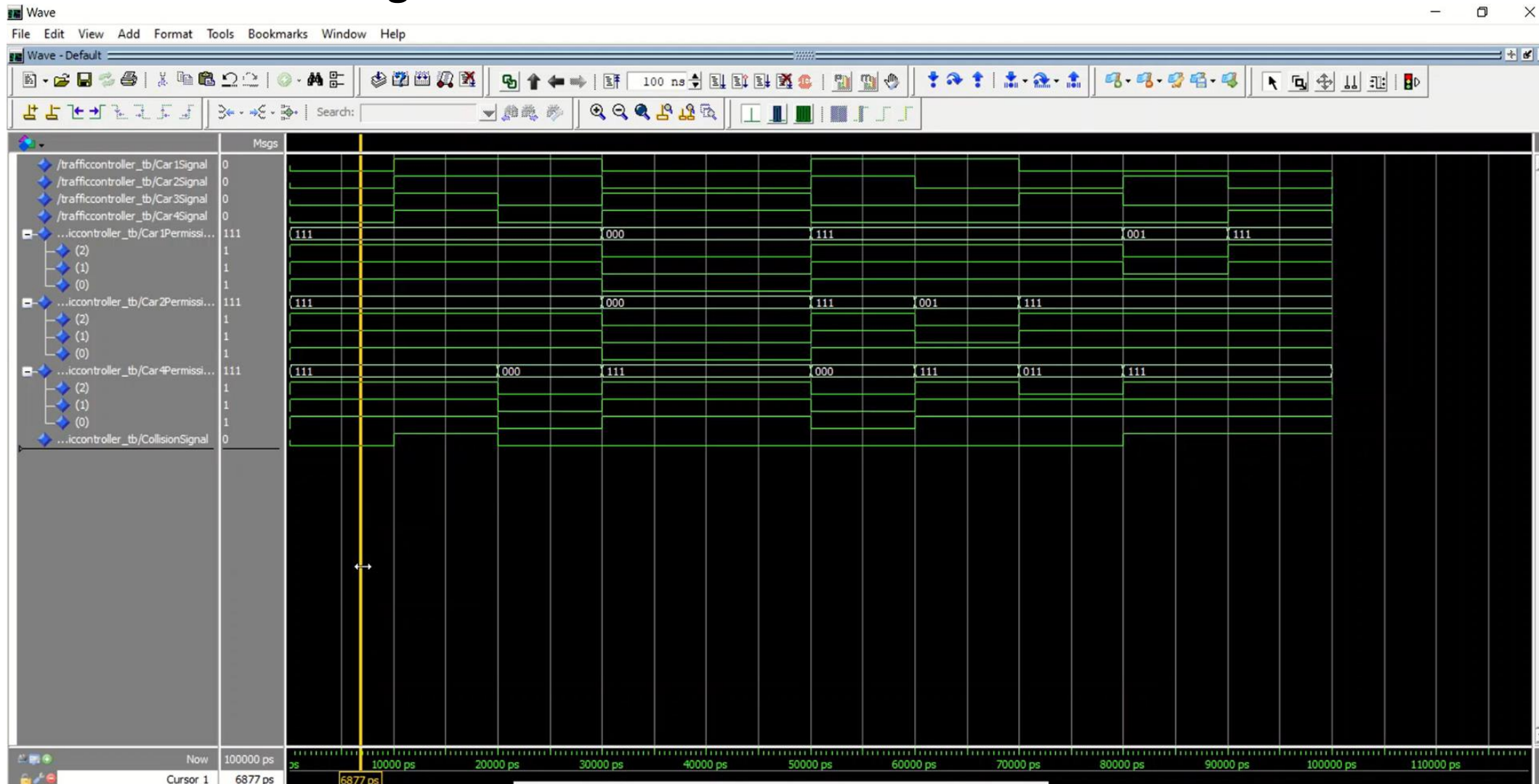
```
-- Car 2 goes right, Car 4 goes right
Car1Signal <= '0';
Car2Signal <= '1';
Car3Signal <= '0';
Car4Signal <= '1';

wait for 10 ns;
```

```
-- Car 1 goes left, Car 2 goes right,
|-- Car 3 goes straight, Car 4 goes right
Car1Signal <= '0';
Car2Signal <= '1';
Car3Signal <= '1';
Car4Signal <= '1';
```

# Hw/Sw Codesign Implementation

- VHDL Simulation using Modelsim



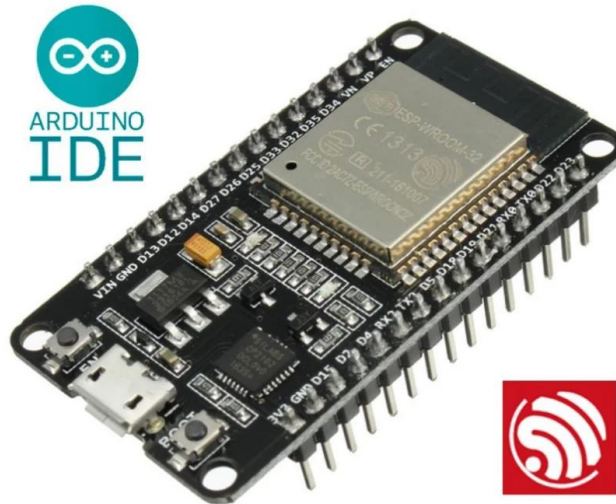
# FreeRTOS Implementation

- Solution approach



[2]

+



[3]

# FreeRTOS Implementations

- **Why ESP-IDF?**

## Advantages

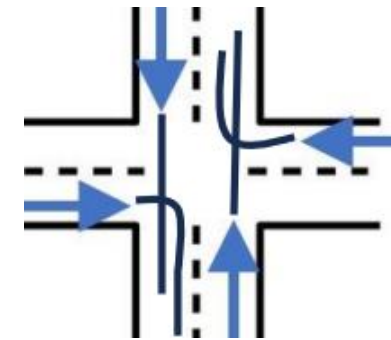
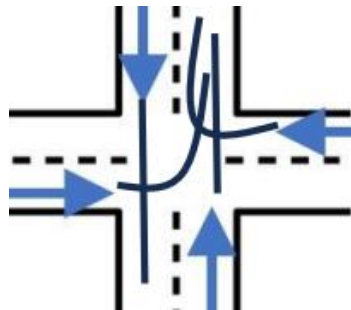
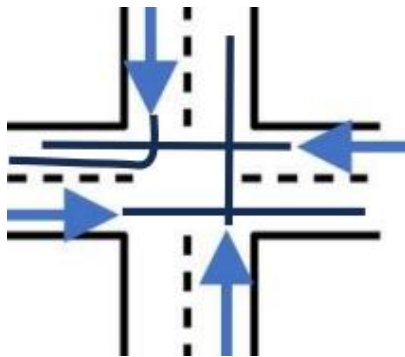
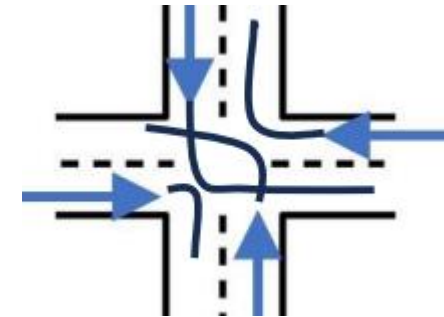
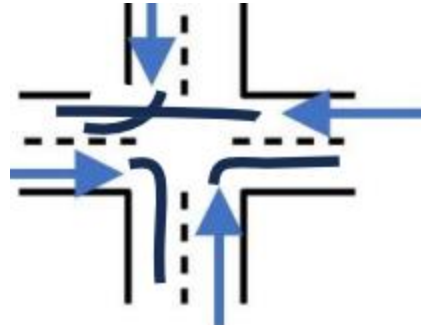
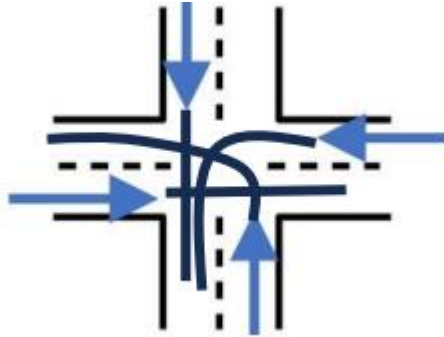
- Enhanced Functionality (library, tools, APIs)
- Access to Advanced Features (Wi-Fi, Bluetooth)
- Rich Documentation and Community Support

## Drawbacks

- Complexity and Development Time
- Hardware-Specific (Espressif's ESP32 and ESP8266)

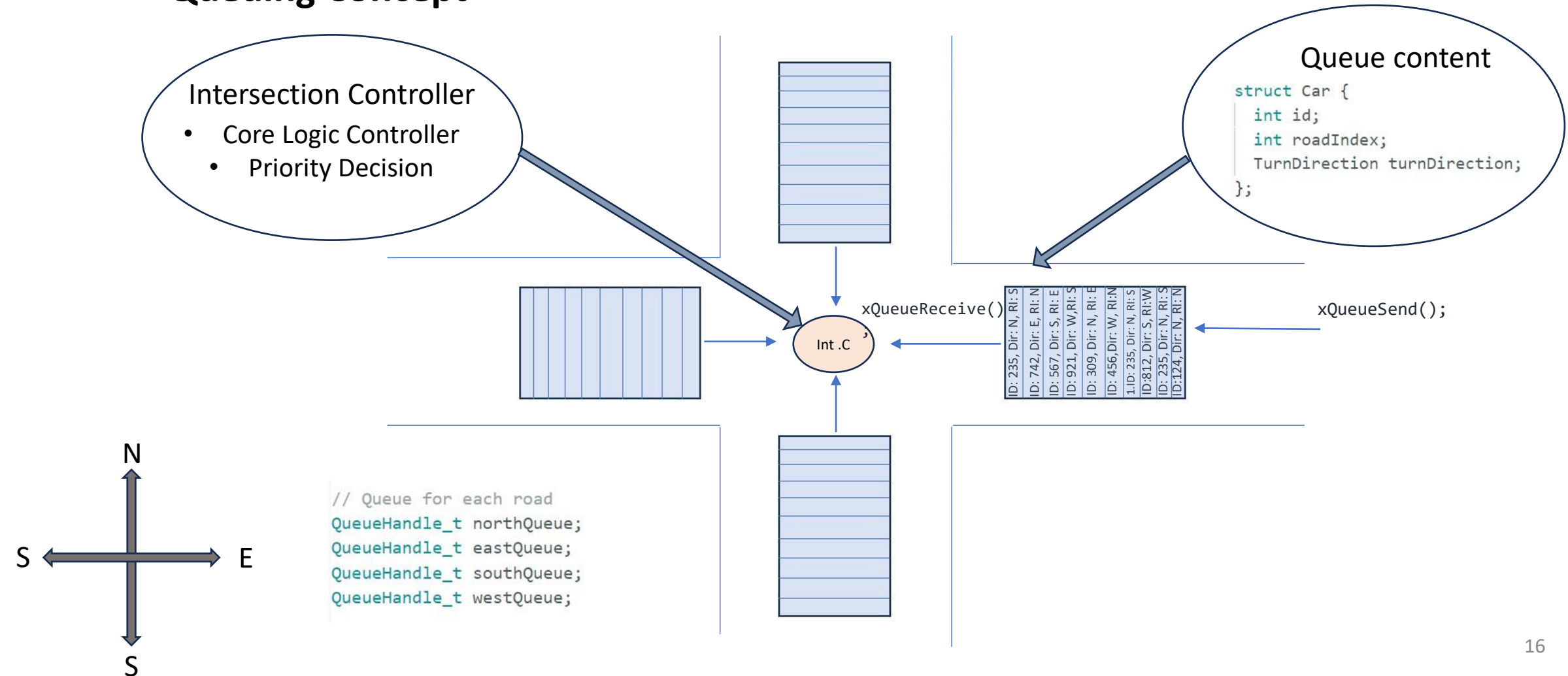
# FreeRTOS Implementations

- Problematic Scenarios



# FreeRTOS Implementations

- Queuing Concept





# FreeRTOS Implementations

- Snippets of code

## Semaphores creation

```
carPassedMutex = xSemaphoreCreateMutex();  
deadlockSemaphore = xSemaphoreCreateBinary();
```

## Queues creation

```
// Create the queues for each road  
northQueue = xQueueCreate(10, sizeof(Car));  
eastQueue = xQueueCreate(10, sizeof(Car));  
southQueue = xQueueCreate(10, sizeof(Car));  
westQueue = xQueueCreate(10, sizeof(Car));
```

## Tasks Creation

```
// Create the intersection controller task  
xTaskCreatePinnedToCore(intersectionControllerTask, "IntersectionControllerTask", 4096, NULL, 1, NULL, tskNO_AFFINITY);  
xTaskCreatePinnedToCore(fillCarIdDirToQueues, "fillQueues", 4096, NULL, 1, NULL, tskNO_AFFINITY);  
// Create the deadlock detection task and schedule it to run on Core 0  
xTaskCreatePinnedToCore(deadlockDetectionTask, "DeadlockDetectionTask", 4096, NULL, 1, NULL, tskNO_AFFINITY);
```

# FreeRTOS Implementations

- Snippets of code

## Sending to the queue

```
// Cars approaching from the North road
if (uxQueueSpacesAvailable(northQueue) > 0) {
    Car northCar;
    northCar.id = random(1000);
    northCar.roadIndex = 0; // North road index
    northCar.turnDirection = static_cast<TurnDirection>(random(1, 4));
    xQueueSend(northQueue, &northCar, portMAX_DELAY);
}
```

## Binary Semaphore

```
xSemaphoreTake(carPassedMutex, portMAX_DELAY);
carPassed = true;
xSemaphoreGive(carPassedMutex);
```

```
void intersectionControllerTask(void* parameter) {
    int count = 0;

    while (true) {
        // Check the queues and allow cars to pass based on booking order
        Car car;

        if (xQueueReceive(northQueue, &car, 0) == pdTRUE) {
            // Check for a matching scenario with a car approaching from the South road
            Car oppositeCar;
            if (xQueueReceive(southQueue, &oppositeCar, 0) == pdTRUE && car.turnDirection == oppositeCar.turnDirection)
                continue;
        }

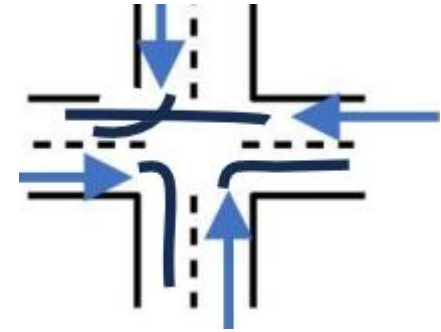
        // ... (rest of the task logic) ...
    }
}
```

## Reading from queue

# FreeRTOS Implemetations

- Results

Car ID: 432, Approaching from:--> North road, Wants to turn:---> Right  
Car ID: 773, Approaching from:--> East road, Wants to turn:---> Straight  
Car ID: 756, Approaching from:--> South road, Wants to turn:---> Right  
Car ID: 940, Approaching from:--> West road, Wants to turn:---> Right



1. Car ID: 432 and 756 are both allowed to pass simultaneously (Both going Straight / turning right).

---

Normal Sequence : North\_Car is allowed to cross the junction -> East\_Car,-> Southast\_Car,-> West\_Car in sequence.

# FreeRTOS Implemetations


- Running FreeRTOS on Hardware

File Edit Sketch Tools Help

ESP32 Wrover Module

car\_1.ino

```
1 #include <Arduino.h>
2 #include <freertos/FreeRTOS.h>
3 #include <freertos/task.h>
4 #include <freertos/queue.h>
5 #include <freertos/semphr.h>
6
7 // Define the turn direction options
8 enum TurnDirection {
9     LEFT,
10    STRAIGHT,
11    RIGHT
12 };
13
14 // Define the road names
15 const char* roadNames[] = {"North", "East", "South", "West"};
16 SemaphoreHandle_t deadlockSemaphore;
17 SemaphoreHandle_t carPassedMutex;
18
19 // Structure to represent a car
20 struct Car {
21     int id;
22     int roadIndex;
23     TurnDirection turnDirection;
24 };
25
26 // Queue for each road
27 QueueHandle_t northQueue;
28 QueueHandle_t eastQueue;
29 QueueHandle_t southQueue;
30 QueueHandle_t westQueue;
31
32 // Flag to track if a car has been allowed to pass
33 bool carPassed = false;
```



Output Serial Monitor

Sketch uses 269082 bytes (94%) of program storage space. Maximum is 1310720 bytes.

Wiedergabe (STRG+P)

Ln 282, Col 42 ESP32 Wrover Module on COM3

00:00:16 00:01:42



# Conclusion

---

# Questions ?

---

# References

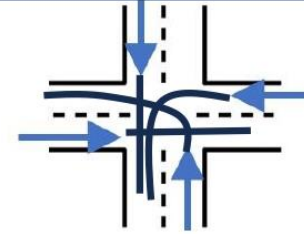
- [1] S. V. U. Feng Zhu, “A linear programming formulation for autonomous intersection control within a dynamic traffic assignment and connected vehicle environment,” Transportation Research Part C, no. 55, p. 363–378, 2015
- [2] <https://www.freertos.org/index.html>
- [3] <https://www.instructables.com/Installing-the-ESP32-Board-in-Arduino-IDE-Windows-/>



# Different Scenarios on Serial Outputs 01

1

Car ID: 569, Approaching from:--> North road, Wants to turn:---> Straight  
Car ID: 700, Approaching from:--> East road, Wants to turn:---> Left  
Car ID: 879, Approaching from:--> South road, Wants to turn:---> Left  
Car ID: 336, Approaching from:--> West road, Wants to turn:---> Straight

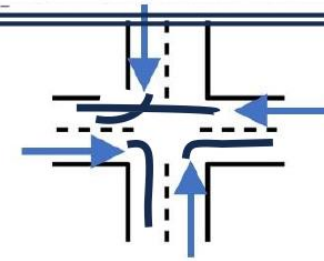


Normal Sequence : North\_Car is allowed to cross the junction -> East\_Car,-> Southast\_Car,-> West\_Car in sequence.

Normal Sequence : North\_Car is allowed to cross the junction -> East\_Car,-> Southast\_Car,-> West\_Car in sequence.

2

Car ID: 432, Approaching from:--> North road, Wants to turn:---> Right  
Car ID: 773, Approaching from:--> East road, Wants to turn:---> Straight  
Car ID: 756, Approaching from:--> South road, Wants to turn:---> Right  
Car ID: 940, Approaching from:--> West road, Wants to turn:---> Right

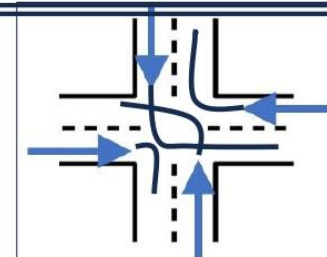


1. Car ID: 432 and 756 are both allowed to pass simultaneously (Both going Straight / turning right).

Normal Sequence : North\_Car is allowed to cross the junction -> East\_Car,-> Southast\_Car,-> West\_Car in sequence.

3

Car ID: 641, Approaching from:--> North road, Wants to turn:--->  
Car ID: 432, Approaching from:--> East road, Wants to turn:---> Right  
Car ID: 412, Approaching from:--> South road, Wants to turn:---> Left  
Car ID: 148, Approaching from:--> West road, Wants to turn:---> Right



Normal Sequence : North\_Car is allowed to cross the junction -> East\_Car,-> Southast\_Car,-> West\_Car in sequence.

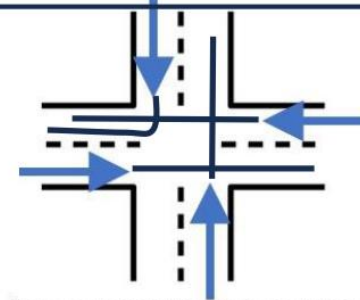
Car ID: 432 and 148 are both allowed to pass simultaneously (Both going Straight / turning right).



# Different Scenarios on Serial Outputs 02

4

Car ID: 342, Approaching from:--> North road, Wants to turn:--->  
Car ID: 22, Approaching from:--> East road, Wants to turn:---> Straight  
Car ID: 410, Approaching from:--> South road, Wants to turn:---> Straight  
Car ID: 739, Approaching from:--> West road, Wants to turn:---> Straight

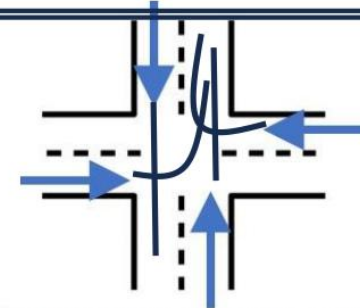


Normal Sequence : North\_Car is allowed to cross the junction -> East\_Car,-> Southast\_Car,-> West\_Car in sequence.

Car ID: 22 and 739 are both allowed to pass simultaneously (Both going Straight / turning right).

5

Car ID: 958, Approaching from:--> North road, Wants to turn:---> Straight  
Car ID: 488, Approaching from:--> East road, Wants to turn:---> Right  
Car ID: 103, Approaching from:--> South road, Wants to turn:---> Straight  
Car ID: 17, Approaching from:--> West road, Wants to turn:---> Left

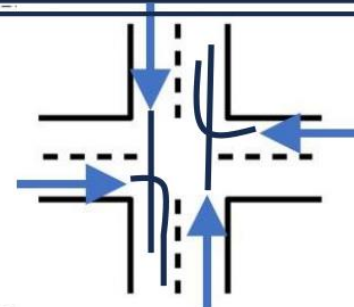


1. Car ID: 958 and 103 are both allowed to pass simultaneously (Both going Straight / turning right).

Normal Sequence : North\_Car is allowed to cross the junction -> East\_Car,-> Southast\_Car,-> West\_Car in sequence.

6

Car ID: 364, Approaching from:--> North road, Wants to turn:---> Straight  
Car ID: 561, Approaching from:--> East road, Wants to turn:---> Right  
Car ID: 827, Approaching from:--> South road, Wants to turn:---> Straight  
Car ID: 942, Approaching from:--> West road, Wants to turn:---> Right



1. Car ID: 364 and 827 are both allowed to pass simultaneously (Both going straight / turning right).

Car ID: 561 and 942 are both allowed to pass simultaneously (Both going Straight / turning right).

# Uppaal Automata

