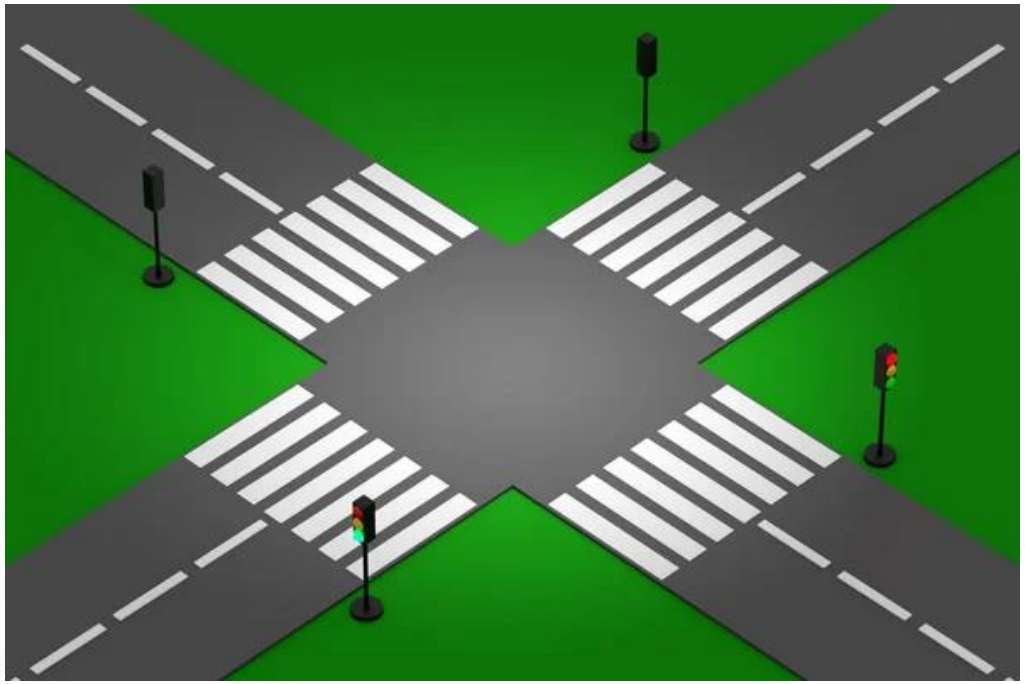
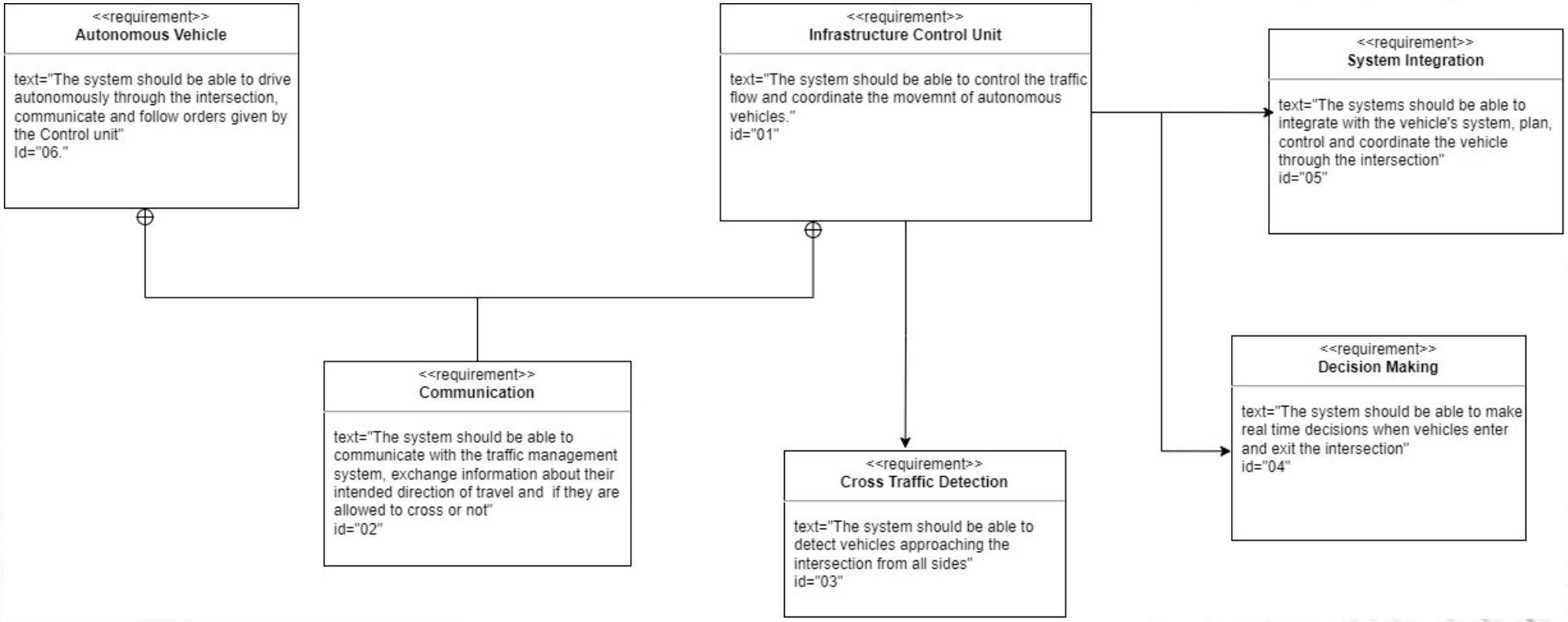
Cross-Traffic Management

Members :
Mehedi Hasan
Masrur Jamil Prochchhod
Pisula Guruge

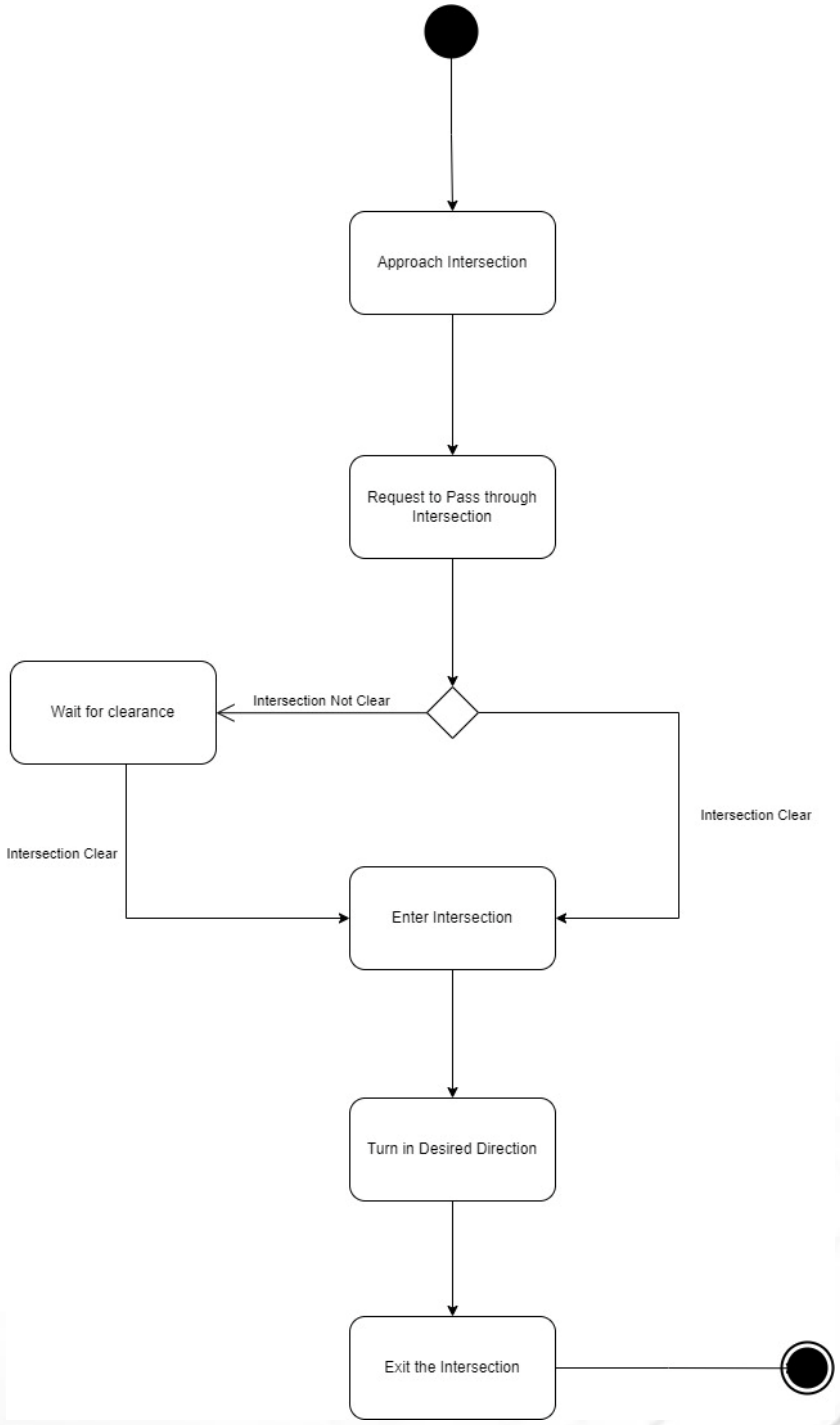
Motivation



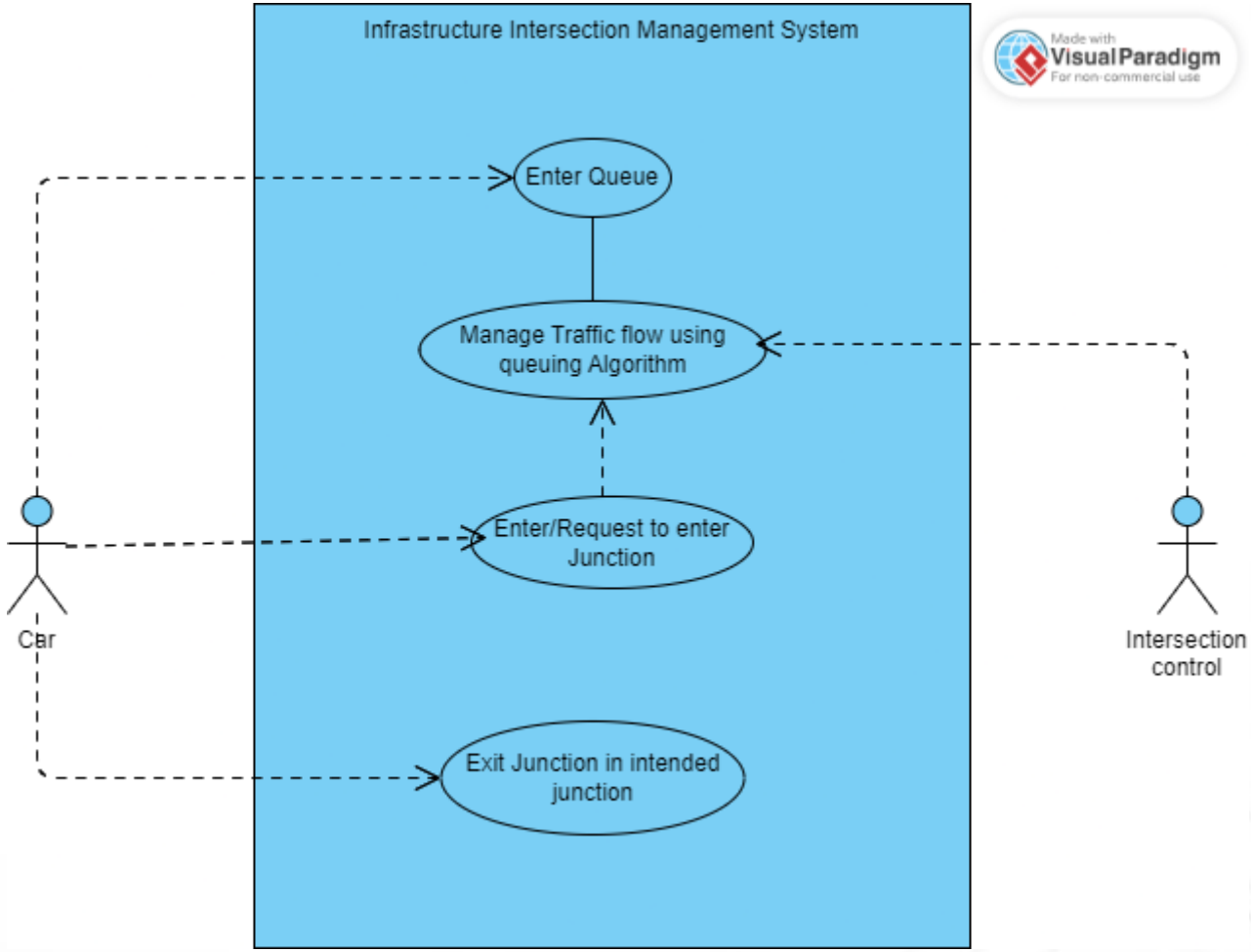
REQUIREMENT DIAGRAMS



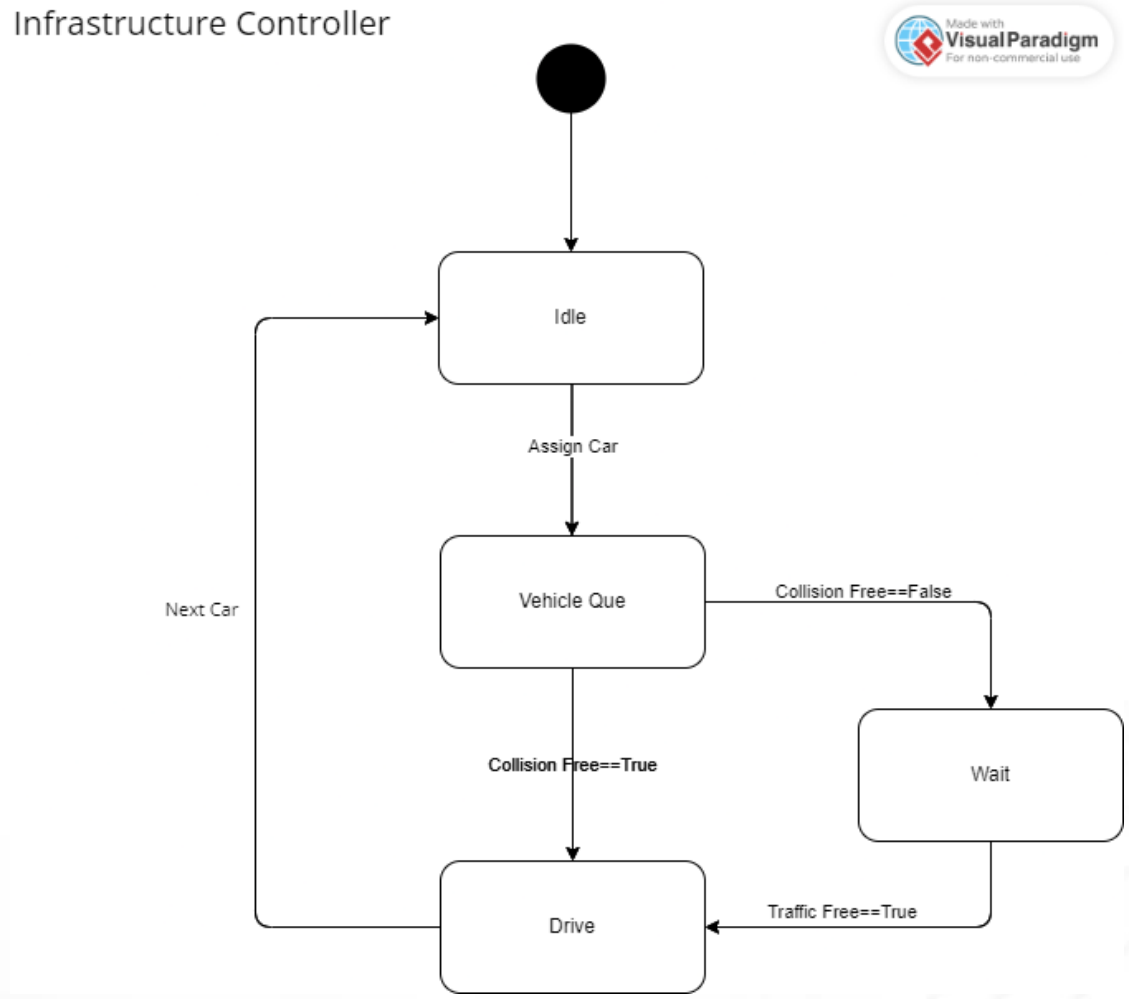
ACTIVITY DIAGRAM



USE CASE DIAGRAM

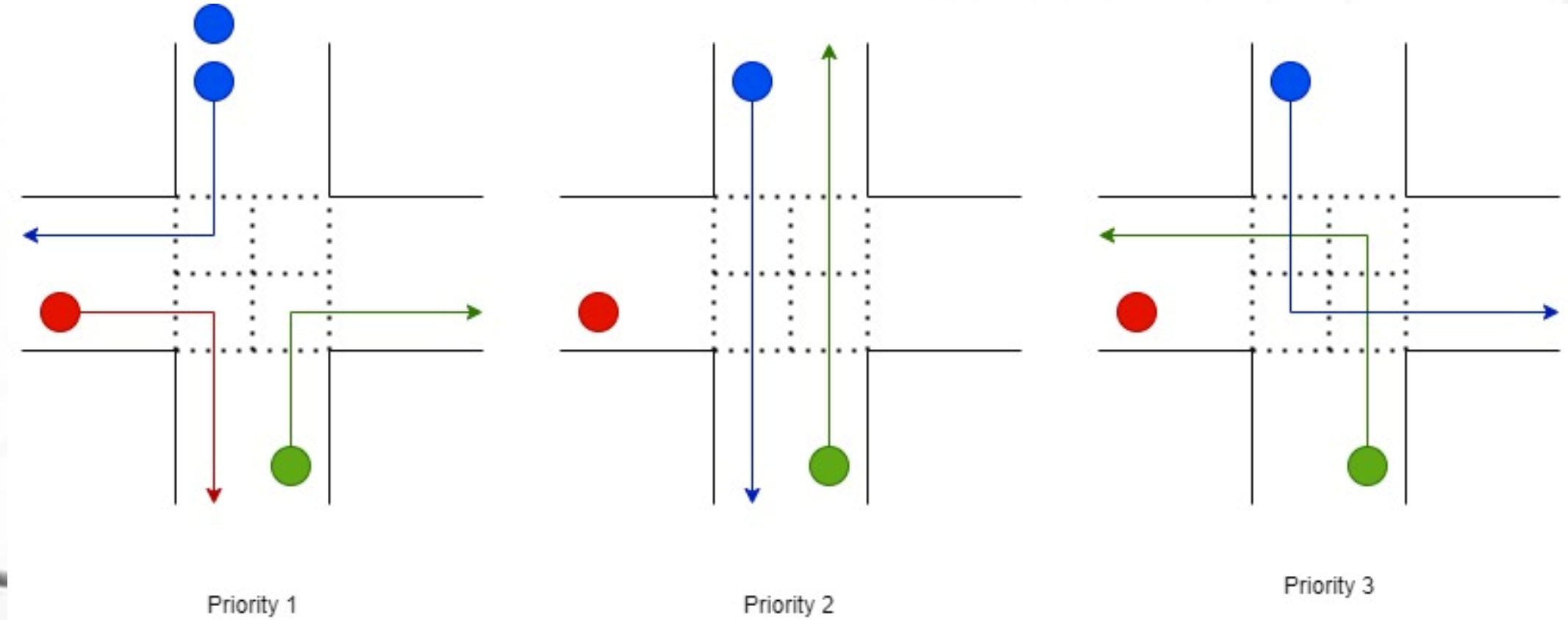


STATE MACHINE DIAGRAM



SCEHDULING ALGORITHM

- 4-way Queuing



IMPLEMENTATION IN C

```c
#include <stdio.h>  // Include standard input-output header
#include <stdlib.h> // Include standard library header for functions like rand() and srand()
#include <time.h>   // Include time header for time() function

#define MAX_VEHICLES 10  // Maximum number of vehicles in a queue
#define NUM_DIRECTIONS 4 // Number of directions (N, S, E, W)

typedef struct {
    int id;         // Vehicle ID
    int speed;      // Speed of the vehicle (1-3 units/time unit)
    char direction; // Direction of the vehicle (N, S, E, W)
} Vehicle; // Define a structure for Vehicle

typedef struct {
    Vehicle vehicles[MAX_VEHICLES]; // Array to store vehicles
    int count;                      // Number of vehicles in the queue
} VehicleQueue; // Define a structure for VehicleQueue
```

IMPLEMENTATION IN C

```c
void addVehicleToQueue(VehicleQueue *queue, Vehicle vehicle) {
    if (!isQueueFull(queue)) { // Check if the queue is not full
        queue->vehicles[queue->count++] = vehicle; // Add vehicle to the queue and increment the count
    } else {
        printf("Queue is full, cannot add vehicle ID %d\n", vehicle.id); // Print message if queue is full
    }
```

```c
Vehicle removeVehicleFromQueue(VehicleQueue *queue) {
    Vehicle emptyVehicle = {-1, 0, ' '}; // Define an empty vehicle to return if queue is empty
    if (!isQueueEmpty(queue)) { // Check if the queue is not empty
        Vehicle vehicle = queue->vehicles[0]; // Get the first vehicle in the queue
```

IMPLEMENTATION IN C

```c
void createVehicle(VehicleQueue *queue, int id) {
    if (isQueueFull(queue)) { // Check if the queue is full
        printf("Queue is full, cannot create vehicle ID %d\n", id); // Print message if queue is full
        return;
    }
    Vehicle newVehicle; // Define a new vehicle
    newVehicle.id = id; // Set the vehicle ID
    newVehicle.speed = rand() % 3 + 1; // Set a random speed between 1 and 3
    newVehicle.direction = possibleDirections[rand() % NUM_DIRECTIONS]; // Set a random direction

    addVehicleToQueue(queue, newVehicle); // Add the new vehicle to the queue
}
```

```c
void simulateTraffic(int totalVehicles) {
    int vehicleCount = 0; // Initialize the vehicle count

    while (vehicleCount < totalVehicles) { // Loop until all vehicles are created
        int direction = rand() % 4; // Randomly choose a direction: 0 = North, 1 = South, 2 = East, 3 = West
```

IMPLEMENTATION IN C

```
113  void roundRobinTrafficControl() {
114      while (1) { // Infinite loop for continuous traffic control
115          if (!isQueueEmpty(&northQueue)) { // Check if the north queue is not empty
116              Vehicle vehicle = removeVehicleFromQueue(&northQueue); // Remove vehicle from the north queue
117              printf("Vehicle ID %d from North queue is crossing the intersection with speed %d\n", vehicle.id, vehicle.speed); // Print vehicle details
118          }
```

FREERTOS

```
85          xTaskCreate(vehicleTask, "VehicleTask", (void *)i, 1, NULL, 0);
86      }
87      xTaskCreate(controlTask, "ControlTask", NULL, 2, NULL, 1);
88  }
```

sketch_jul3b.ino

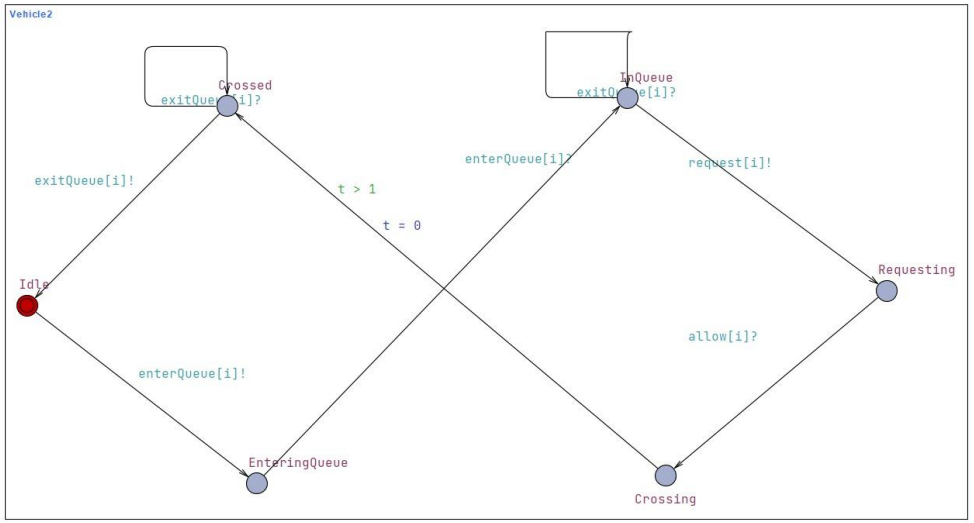
```
1   #include <Arduino.h>
2   #include "freertos/FreeRTOS.h"
3   #include "freertos/task.h"
4   #include "freertos/queue.h"
5   #include "freertos/semphr.h"
6
7   // Define the Vehicle structure
8   struct Vehicle {
9       int id;
10      int speed; // 1-3 units/time unit
11      char direction; // N, S, E, W
12  };
```

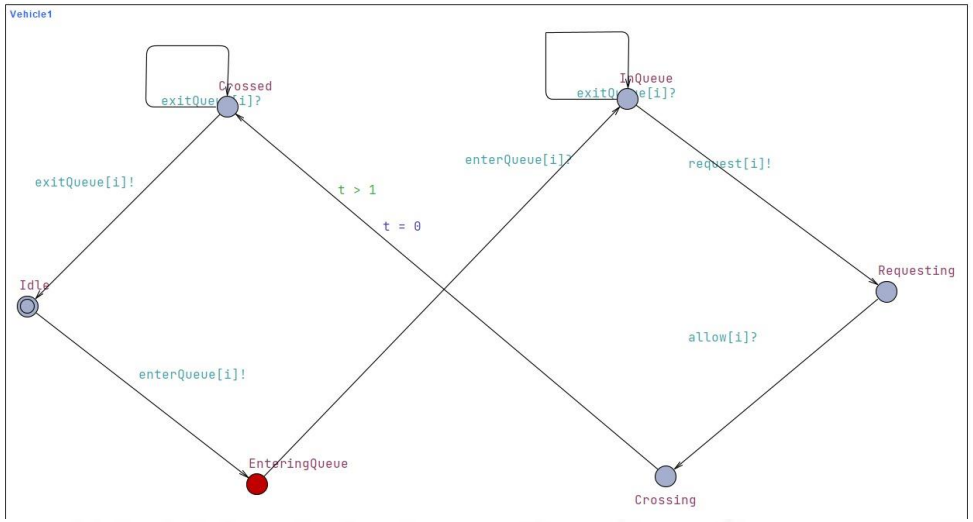
Output Serial Monitor ×

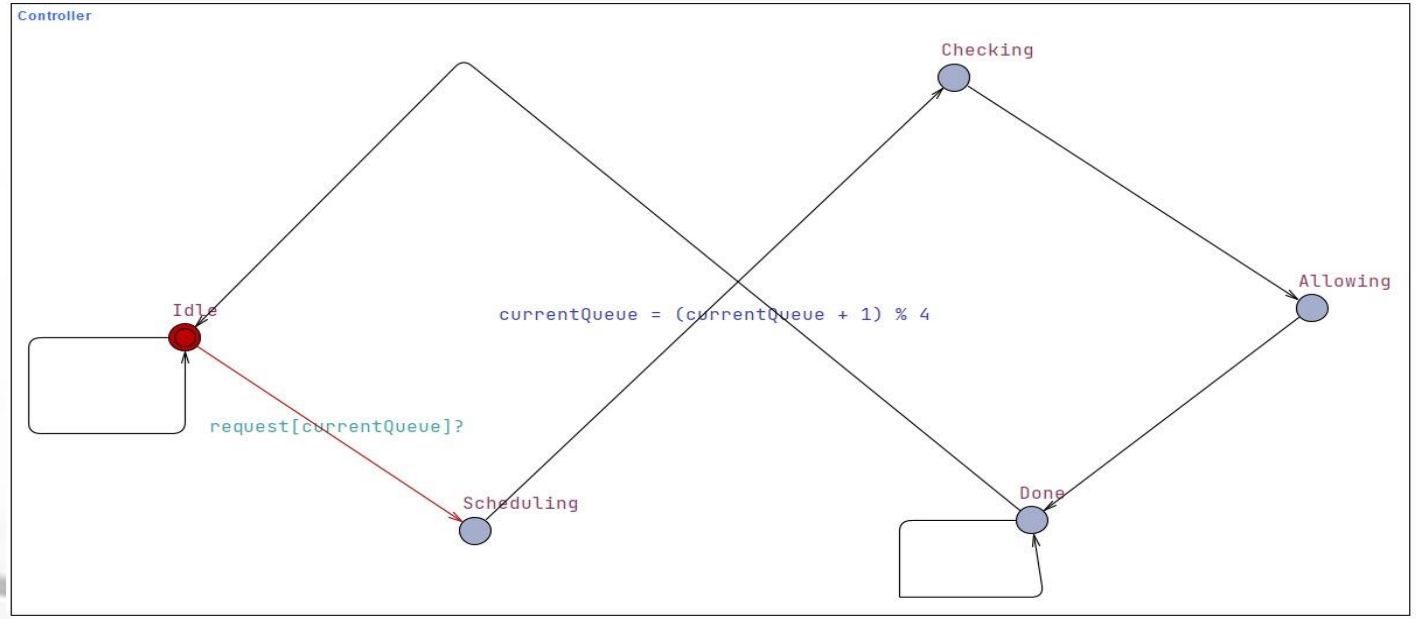
Message (Enter to send message to 'ESP32S3 Dev Module' on 'COM7')

```
`!□□□□□□3□□9e□)Vehicle ID 3 from North queue is crossing the intersection with speed 2
Vehicle ID 4 from South queue is crossing the intersection with speed 1
Vehicle ID 1 from East queue is crossing the intersection with speed 2
Vehicle ID 12 from North queue is crossing the intersection with speed 2
Vehicle ID 14 from South queue is crossing the intersection with speed 1
Vehicle ID 2 from East queue is crossing the intersection with speed 1
Vehicle ID 8 from North queue is crossing the intersection with speed 1
Vehicle ID 6 from South queue is crossing the intersection with speed 3
Vehicle ID 13 from East queue is crossing the intersection with speed 3
Vehicle ID 5 from North queue is crossing the intersection with speed 1
Vehicle ID 9 from East queue is crossing the intersection with speed 2
Vehicle ID 15 from North queue is crossing the intersection with speed 2
Vehicle ID 16 from East queue is crossing the intersection with speed 2
Vehicle ID 17 from North queue is crossing the intersection with speed 3
Vehicle ID 10 from East queue is crossing the intersection with speed 1
Vehicle ID 19 from North queue is crossing the intersection with speed 1
Vehicle ID 18 from East queue is crossing the intersection with speed 2
Vehicle ID 11 from East queue is crossing the intersection with speed 3
Vehicle ID 7 from East queue is crossing the intersection with speed 2
Vehicle ID 20 from East queue is crossing the intersection with speed 3
```

UPPAAL DIAGRAM







VHDL

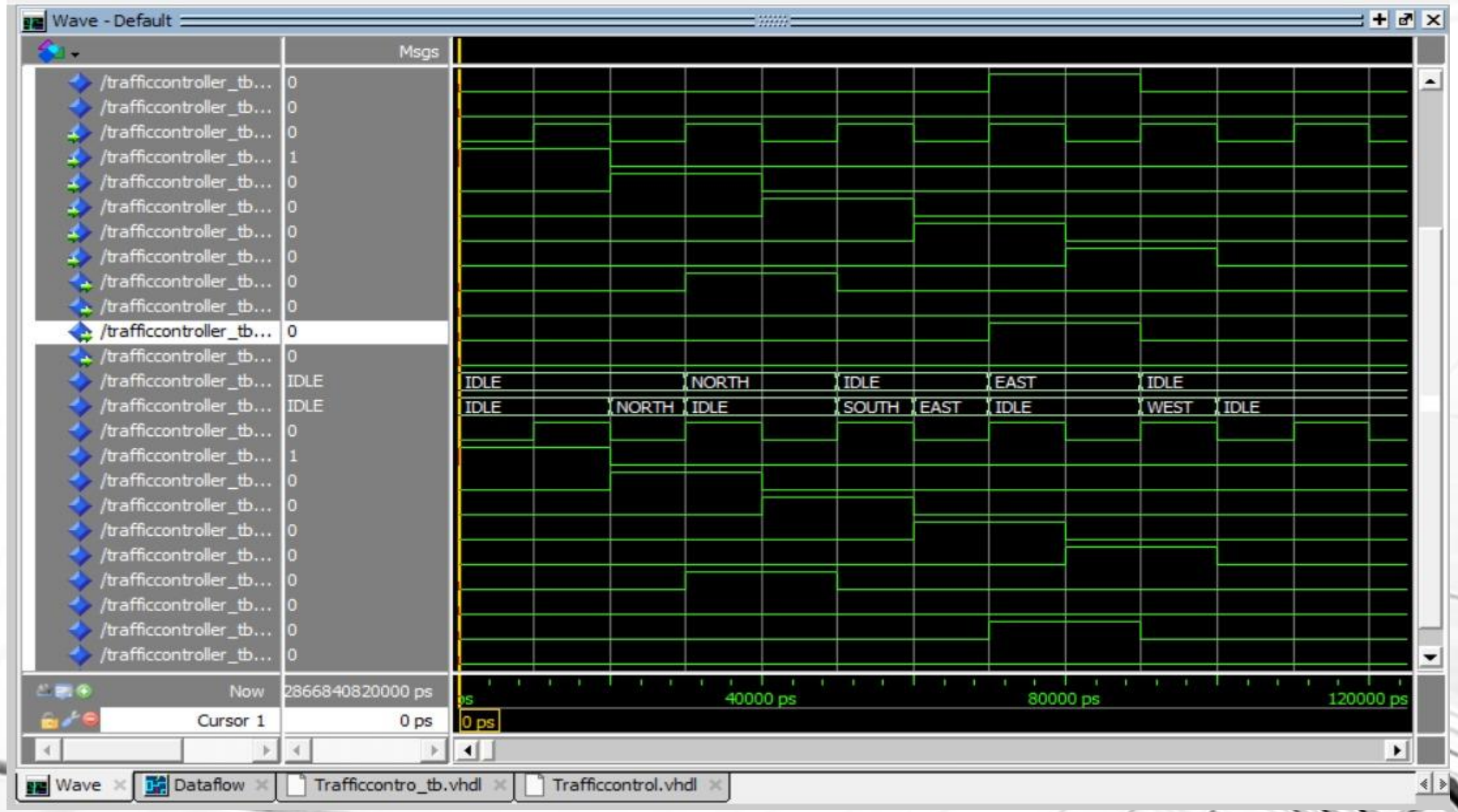
```
21  architecture Behavioral of TrafficController is
22      type state_type is (IDLE, NORTH, SOUTH, EAST, WEST);
23      signal state : state_type := IDLE;
24      signal next_state : state_type;
25
26  begin
27
28      process (clk, reset)
29      begin
30          if reset = '1' then
31              state <= IDLE;
32          elsif rising_edge(clk) then
33              state <= next_state;
34          end if;
35      end process;
```

```
when SOUTH =>
    allow_north <= '0';
    allow_south <= '1';
    allow_east <= '0';
    allow_west <= '0';
    next_state <= IDLE;
```

```
            when others =>
                allow_north <= '0';
                allow_south <= '0';
                allow_east <= '0';
                allow_west <= '0';
                next_state <= IDLE;
        end case;
    end process;

end Behavioral;
```

MODELSIM


Wave - Default
Msgs
/trafficcontroller_tb...
IDLE
NORTH
SOUTH
EAST
WEST
Now
2866840820000 ps
Cursor 1
0 ps
40000 ps
80000 ps
120000 ps
Wave
Dataflow
Trafficcontro_tb.vhdl
Trafficcontrol.vhdl

Thank You