ECEN/CSCE 689 PROJECT

MODEL CHECKING OF TRAFFIC CONTROL SYSTEM WITH VEHICLE SIMULATION

Phase B Report   Task B1: Integration

TEAM – 10

V-GROUP

SUBMITTED BY

MEGHANA JAYSING AMUP (632000331)

RISHYA SANKAR KUMARAN (931002051)

Introduction

In project phase A, the vehicles and the infrastructure were designed independently. The vehicles generated in phase A have assumed the signal information and positions were updated accordingly. Similarly, the infrastructure has assumed the vehicle positions and controlling the signal lights for each junction in the system. In phase B, the vehicle and infrastructure implementations are integrated to build the entire system. Hence, the vehicles will update their positions and move to through the grid according to the actual signals provided by infrastructure and the signals will be switched according to the current congestion information of the system at the particular time instance.
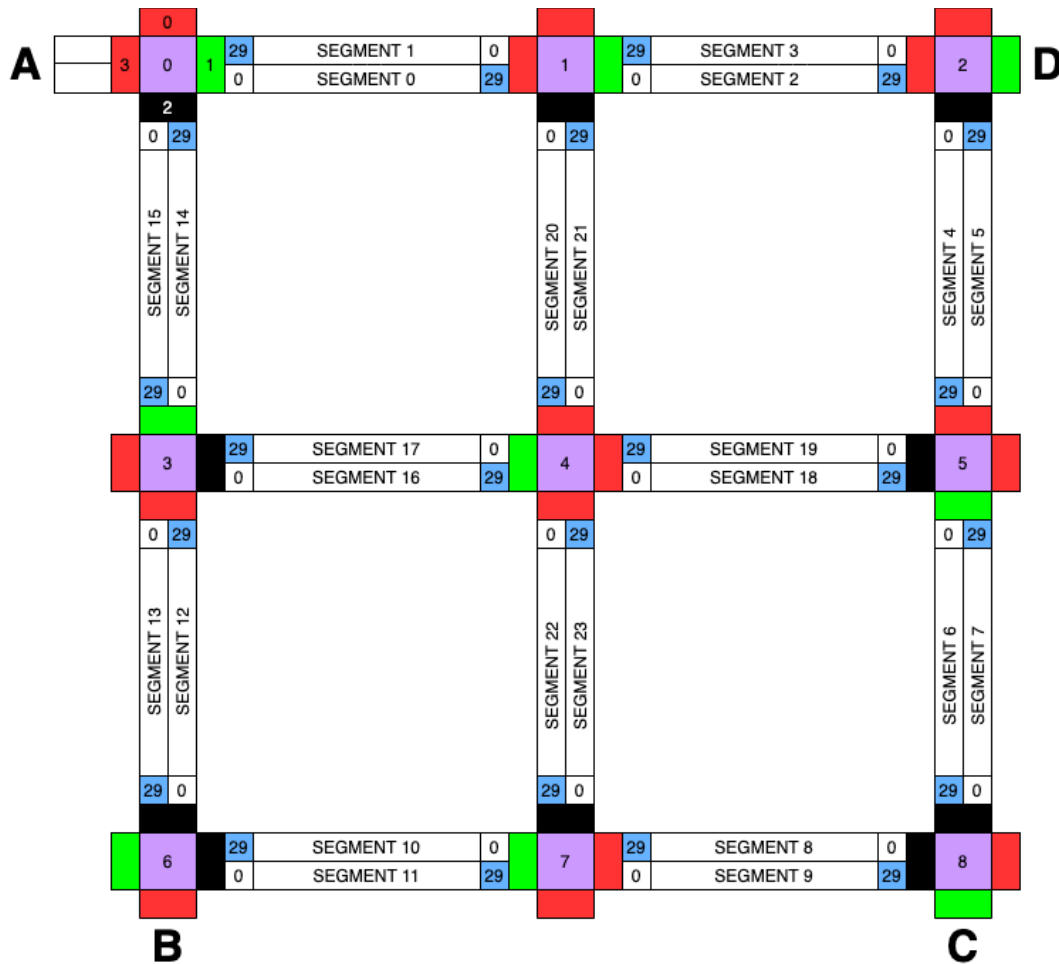
The road system described in the given problem statement is visualized as shown in the Figure above.

Algorithm for Integrated System
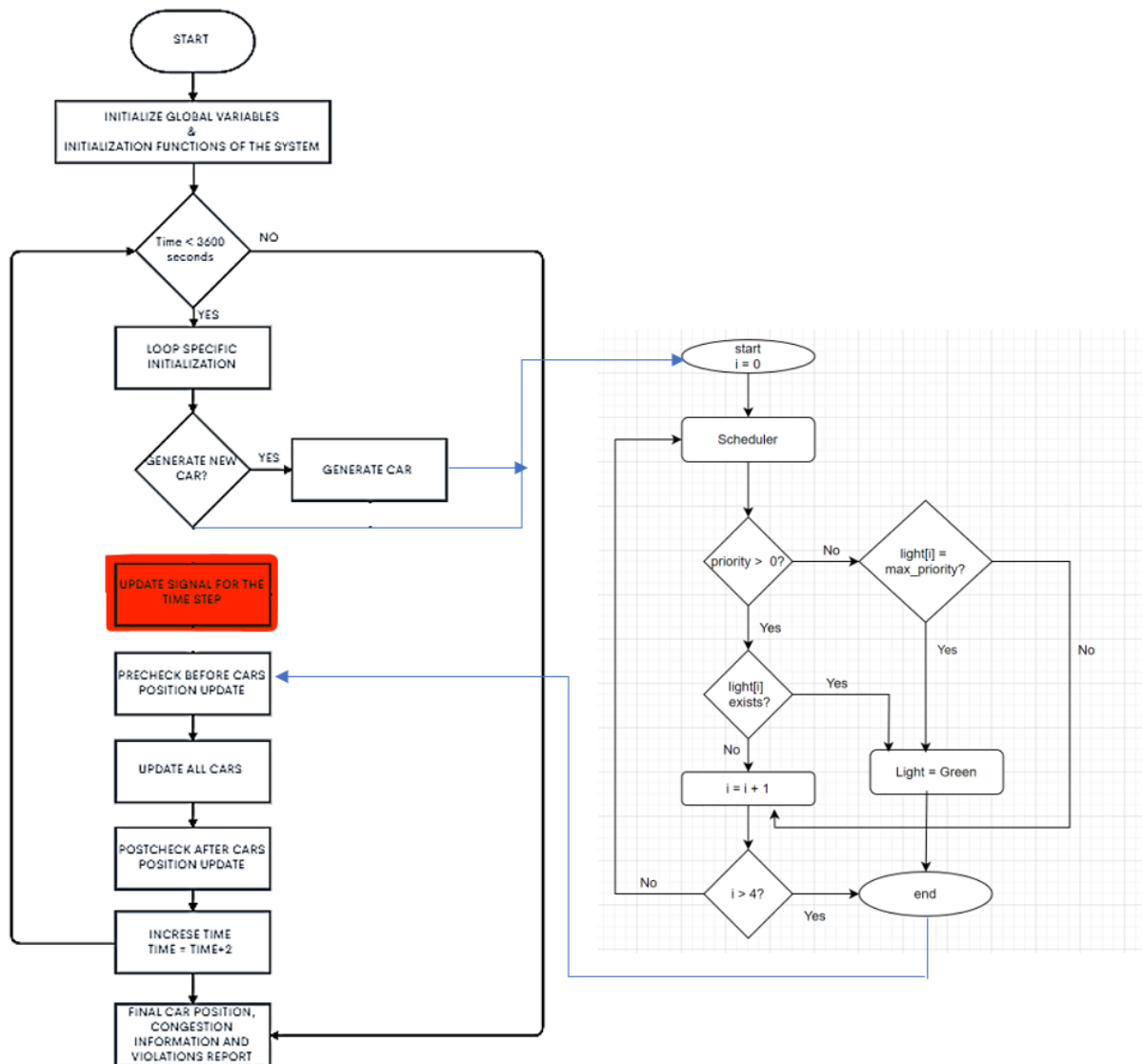
The left flowchart is the implementation of vehicle and the right flowchart is the implementation of infrastructure. The above diagram is the flowchart for the entire integrated traffic system.

System Overview

Infrastructure Implementation	Vehicle Implementation

Variables:
 	
```int t```
```int segment[24][30]```
```int il[9][4]```
(Above variables are shared between vehicle and infrastructure implementation)
 	
```std::vector<std::queue<int>> q(9)```
```int signal[24][2]```
```int p_light[NUM_JUNCTIONS][NUM_SIGNALS]```
```int segment_info[NUM_JUNCTIONS][NUM_SIGNALS]```
	Variables:
	
	```int t```
	```int segment[24][30]```
	```int il[9][4]```
	(Above variables are shared between vehicle and infrastructure implementation)
	
	```int *route[19]```
	```int *car[900]```
	```int total_completed_cars```
	```unsigned int total_cars```
	```int total_crash```
	```int junction0_prech[2];```
	```int junction0_posch[2];```
	```int u_turn_vio = 0;```
	```int r_vio = 0;```
	```int mis_dir_cnt = 0;```

Functions:
 	
```void initial_signal_priority()```
```void priority_signal()```
```int get_car_count_from_segment (int segmentno)```
```int check_priority (int num_cars[])```
```void scheduler (int jun[], int priority, int pri[], int first, int num_cars[])```
```void controller (int time)```
```void initialise_queue ()```
	Functions:
	
	```void create_possible_routes()```
	```void generate_car(int id)```
	```void update_car(int id)```
	```bool check_crash_j0()```
	```int collision_in_a_segment(int exp_total, int act_total)```
	```int check_utrun_vio()```
	```int check_r_vio()```
	```void create_segments();```
	```void filled_slot_info(int t);```

The variable highlighted in yellow was assumed by I- group before integration and will be taken from v-group now. Similarly, The variable highlighted in green was assumed by V- group before integration and will be taken from I-group now.

Challenges faced during integration

1. The challenge was to understand the implementation of infrastructure team. The understanding of the different functions was essential to integrate the vehicle and infrastructure implementation.
2. During integration, we needed to adopt to the format of signal information shared by infrastructure team.
3. One of the assumptions of the infrastructure team was different from the vehicle team.
4. In the main function, the vehicles were generated and position was updated in a while loop. In the same while loop, the signal switching implementation was added. The critical part was sequence of updating the signals and generating cars and updating positions.
5. Some bugs were discovered in the system after integration.
   For instance, below are the two cases:
   a. One vehicle was stuck at the last slot of the last segment before returning to the destination.
   b. While switching signals, the queue for a signal was not updated.
   c. For one junction, the signal was not switching according to the designed priority algorithm.

How the challenges were overcome

1. We studied the phase A report of the infrastructure group. After that we had the discussions with the infrastructure group to understand few specific functions.
2. The vehicle position and signal states at each junction were the shared information used by the system. The information format was analyzed and then necessary changes were made to integrate the system.
3. To fix the problem caused due to different assumption was resolved by considering one assumption an modifying the code according to the same so that whole system will follow as assumptions.
4. The sequence was implemented in the main function to update the vehicles according to signals and time. Both the signal switching and vehicle position updating are synchronized w.r.t time.
5. The integration helped to identify bugs in the system because most of the test cases were covered when simulation was run with signals switching for every time step.
   a. One vehicle was stuck at the last slot of the last segment before returning to the destination because the route information was wrong due to the typo.
   b. The switching signal queue was updated in the flow of the code.
   c. The mapping was incorrect for the junction where the signal was not switching according to the designed priority algorithm. The mapping was the signal was updated.
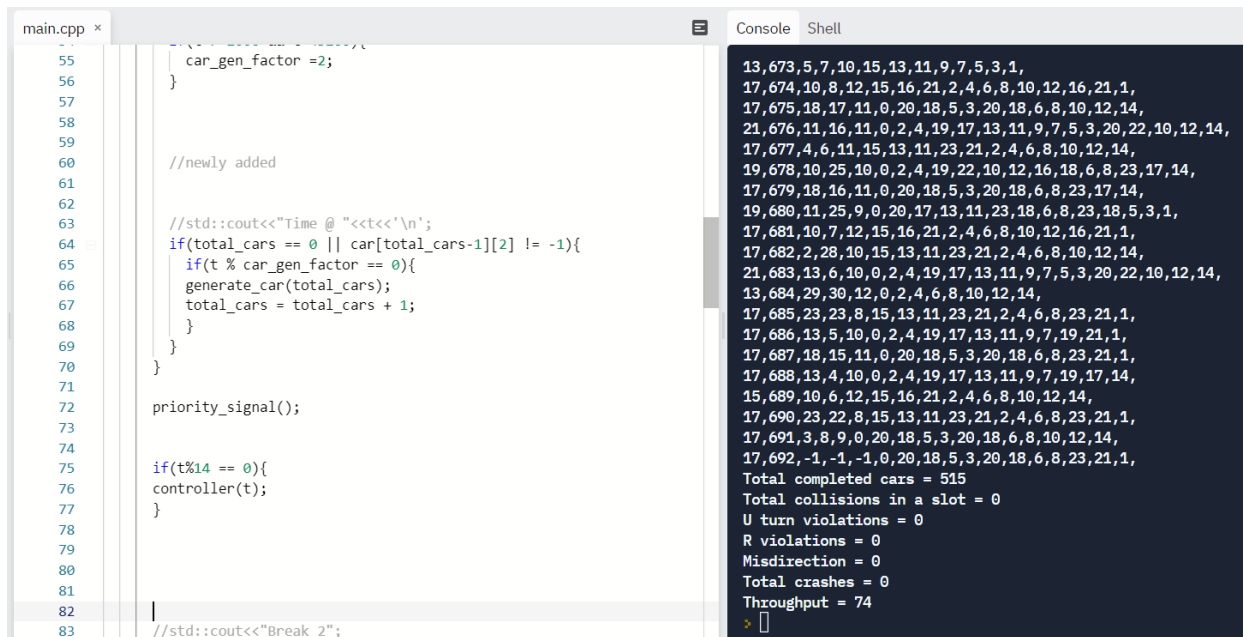
Results

1. Output of the integrated system

The below is snippet of the console providing car positions and Signal information:

```
lights at time t = 1078
intersection = 0
2 1 0 1
intersection = 1
1 2 0 1
intersection = 2
1 2 0 0
intersection = 3
1 0 1 2
intersection = 4
1 1 2 1
intersection = 5
1 1 2 0
intersection = 6
0 0 2 1
intersection = 7
0 1 1 2
intersection = 8
0 1 2 0
@Time : 1078
0,365,364,0,0,0,360,359,0,0,0,0,0,0,0,0,357,0,355,0,353,0,0,0,0,0,0,350,349,348,
0,0,0,0,0,0,0,28,0,0,0,0,0,0,0,0,0,170,0,0,26,0,0,0,0,58,45,43,29,
0,0,0,0,267,265,0,0,0,0,0,0,0,0,259,0,0,0,0,0,0,0,0,0,0,0,0,0,229,0,
0,0,0,0,0,0,287,285,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,177,175,53,46,38,0,
345,0,0,0,0,0,0,343,342,340,339,227,225,223,221,293,281,270,261,256,0,332,331,209,202,194,186,0,327,326,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,61,60,56,0,206,0,
0,181,179,172,166,162,155,149,0,214,208,200,183,178,0,317,312,311,310,308,148,135,122,121,120,119,117,0,176,173,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,102,93,92,0,224,88,83,0,0,82,81,220,218,212,211,
137,116,107,104,103,99,95,94,91,85,84,74,73,70,69,59,57,49,47,0,161,160,158,54,0,255,254,251,243,241,
0,0,0,0,0,289,205,204,198,195,188,187,185,184,182,174,169,272,164,152,141,140,138,263,136,133,131,125,123,113,
0,0,147,142,0,0,0,232,0,0,0,0,0,0,0,0,274,215,210,201,0,129,0,0,0,0,0,0,124,109,
329,0,0,0,0,0,0,253,252,250,249,248,245,242,321,320,319,318,315,313,309,306,240,237,236,231,228,226,219,207,
0,0,0,146,127,115,111,108,90,52,44,0,247,64,5,0,0,0,0,0,0,0,244,42,0,233,14,3,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,351,0,0,0,0,0,0,0,266,264,262,0,337,336,335,334,330,
19,0,217,203,196,87,79,77,0,0,0,0,0,0,0,66,0,193,192,190,189,1,41,15,167,25,17,7,0,165,
0,0,0,363,362,361,0,0,0,0,0,0,0,0,0,358,0,356,0,354,0,352,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,347,0,62,0,0,0,0,0,0,341,333,328,322,112,39,34,
0,106,68,0,0,0,0,0,0,0,0,0,0,0,97,0,0,0,0,0,0,0,0,96,89,78,30,0,269,268,
292,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,199,168,163,71,
325,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,75,67,314,307,305,301,55,35,299,295,291,286,283,
346,0,0,0,0,0,0,0,0,0,0,0,0,0,280,276,32,31,344,338,324,323,316,238,230,222,216,0,303,296,
0,0,0,65,0,105,0,282,278,275,0,0,0,98,0,2,0,304,302,300,298,294,273,0,271,0,0,0,33,0,
0,0,0,0,0,0,0,0,0,0,0,0,279,277,260,257,246,239,235,234,213,197,191,180,171,159,153,151,128,118,
157,156,0,0,134,48,0,0,297,290,0,288,284,0,0,0,0,0,0,0,0,130,0,0,258,0,0,0,0,0,0,
```

2. Sample output after integration that shows all the violations are 0

```
main.cpp ×
55        car_gen_factor =2;
56      }
57
58
59
60      //newly added
61
62
63      //std::cout<<"Time @ "<<t<<'\n';
64      if(total_cars == 0 || car[total_cars-1][2] != -1){
65        if(t % car_gen_factor == 0){
66        generate_car(total_cars);
67        total_cars = total_cars + 1;
68        }
69      }
70    }
71
72    priority_signal();
73
74
75    if(t%14 == 0){
76    controller(t);
77    }
78
79
80
81
82    |
83    //std::cout<<"Break 2";
```

```
Console  Shell
13,673,5,7,10,15,13,11,9,7,5,3,1,
17,674,10,8,12,15,16,21,2,4,6,8,10,12,16,21,1,
17,675,18,17,11,0,20,18,5,3,20,18,6,8,10,12,14,
21,676,11,16,11,0,2,4,19,17,13,11,9,7,5,3,20,22,10,12,14,
17,677,4,6,11,15,13,11,23,21,2,4,6,8,10,12,14,
19,678,10,25,10,0,2,4,19,22,10,12,16,18,6,8,23,17,14,
17,679,18,16,11,0,20,18,5,3,20,18,6,8,23,17,14,
19,680,11,25,9,0,20,17,13,11,23,18,6,8,23,18,5,3,1,
17,681,10,7,12,15,16,21,2,4,6,8,10,12,16,21,1,
17,682,2,28,10,15,13,11,23,21,2,4,6,8,10,12,14,
21,683,13,6,10,0,2,4,19,17,13,11,9,7,5,3,20,22,10,12,14,
13,684,29,30,12,0,2,4,6,8,10,12,14,
17,685,23,23,8,15,13,11,23,21,2,4,6,8,23,21,1,
17,686,13,5,10,0,2,4,19,17,13,11,9,7,19,21,1,
17,687,18,15,11,0,20,18,5,3,20,18,6,8,23,21,1,
17,688,13,4,10,0,2,4,19,17,13,11,9,7,19,17,14,
15,689,10,6,12,15,16,21,2,4,6,8,10,12,14,
17,690,23,22,8,15,13,11,23,21,2,4,6,8,23,21,1,
17,691,3,8,9,0,20,18,5,3,20,18,6,8,10,12,14,
17,692,-1,-1,-1,0,20,18,5,3,20,18,6,8,23,21,1,
Total completed cars = 515
Total collisions in a slot = 0
U turn violations = 0
R violations = 0
Misdirection = 0
Total crashes = 0
Throughput = 74
> ▯
```

In this case, total 692 cars were generated and 515 successfully returned in 1 hour achieving throughput of 74%.
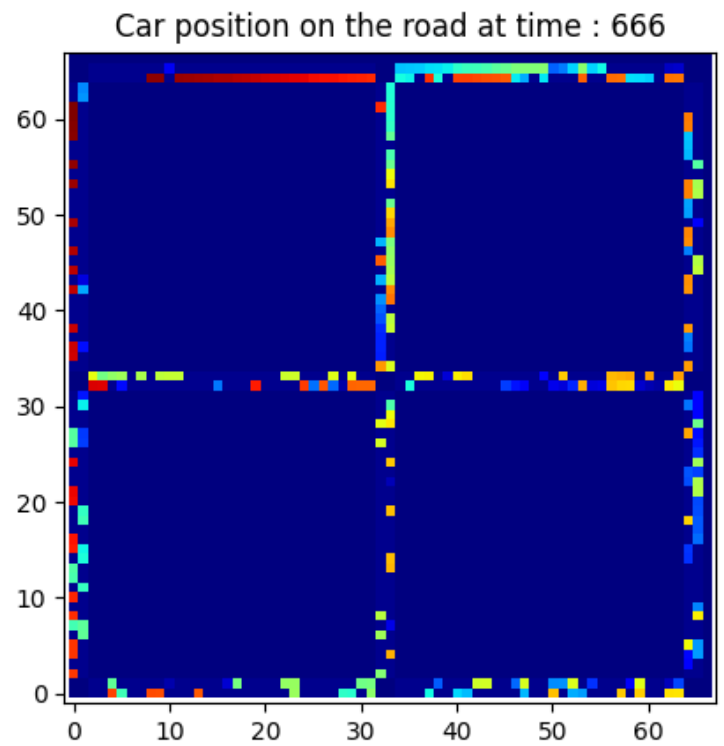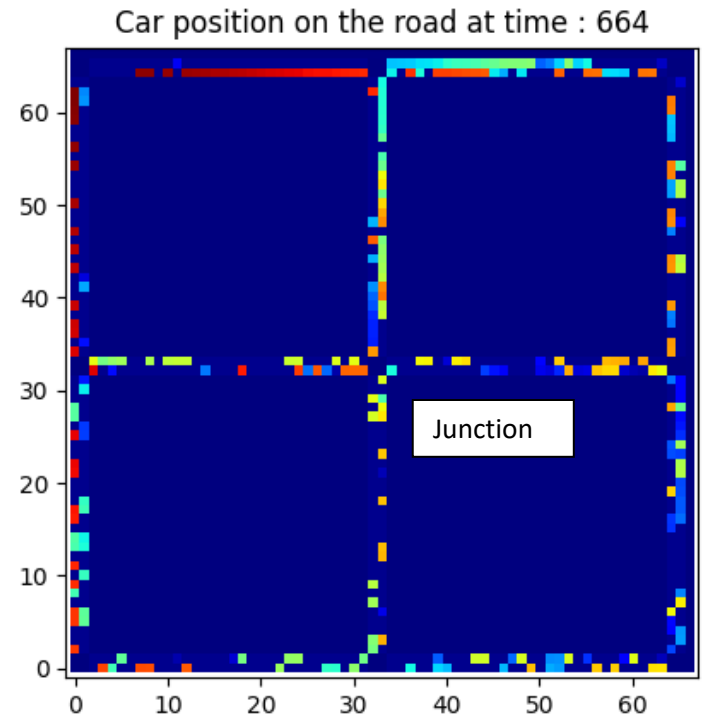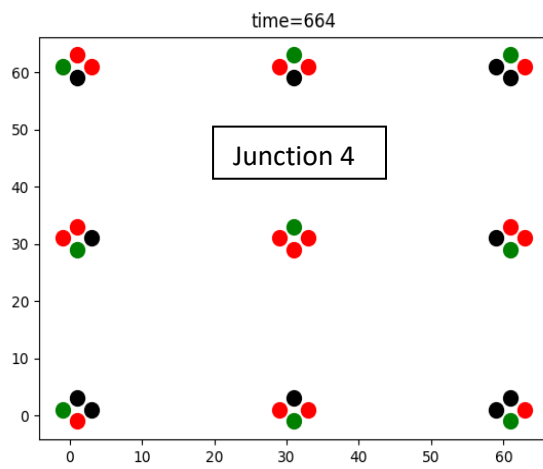There were no red-light violations, U-turn violations and no car were running in wrong direction in the lane (Misdirection). Also, there are no crashes at the intersection (Total crashes) or in a lane (Total collisions in a slot).
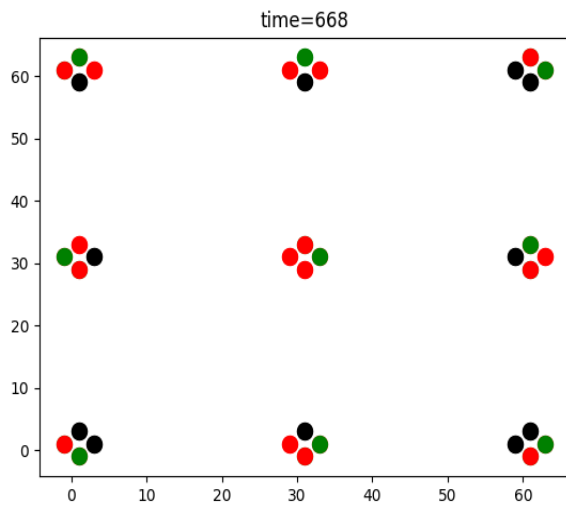
3. Visual representation of Integrated system:

We are going to analyse 1 sample output the time period from t = 664 to t =676 at junction 4 as shown on below diagrams.

At junction 4, at time 664, we can see that top light is Green which indicates that vehicles from the bottom segments should move. It is evident from below snapshots that yellow car from end of bottom segment at 664 has moved to the beginning of the top segment at t = 666. For all other lanes i.e. top, left and right, the cars position is not changed between t = 664 to t =666.

At junction 1, at time 672, we can see that left light is Green which indicates that vehicles from the right segment should move. It is evident from below snapshots that blue car from end of right segment at 672 has moved to the beginning of the bottom segment at t = 674. For all other lanes i.e. top, left and right, the cars position is not changed between t = 670 to t =674.

time=664

Junction 4

Car position on the road at time : 664

Junction

Car position on the road at time : 666

Junction 1

time=668

Car position on the road at time : 668

Junction 1

Car position on the road at time : 670

time=672

Car position on the road at time : 672

Car position on the road at time : 674
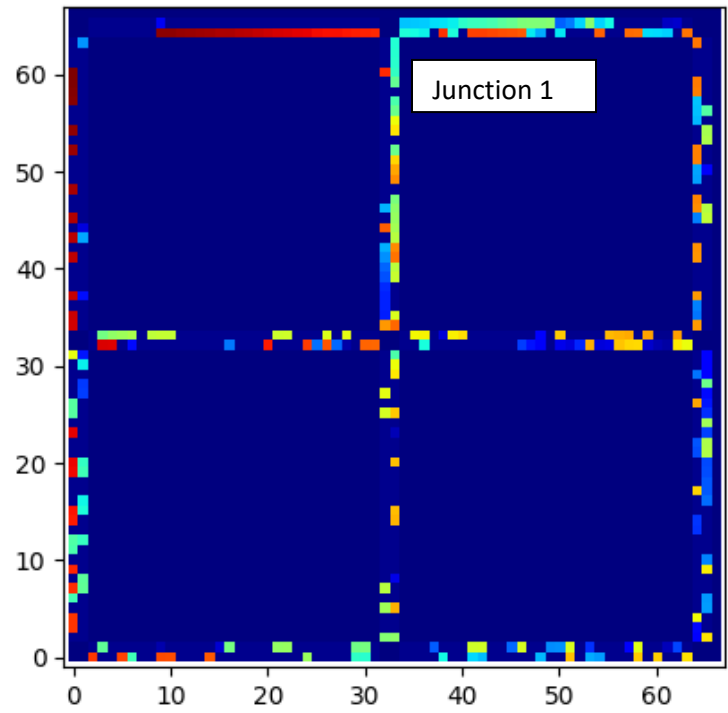
time=676

Car position on the road at time : 676
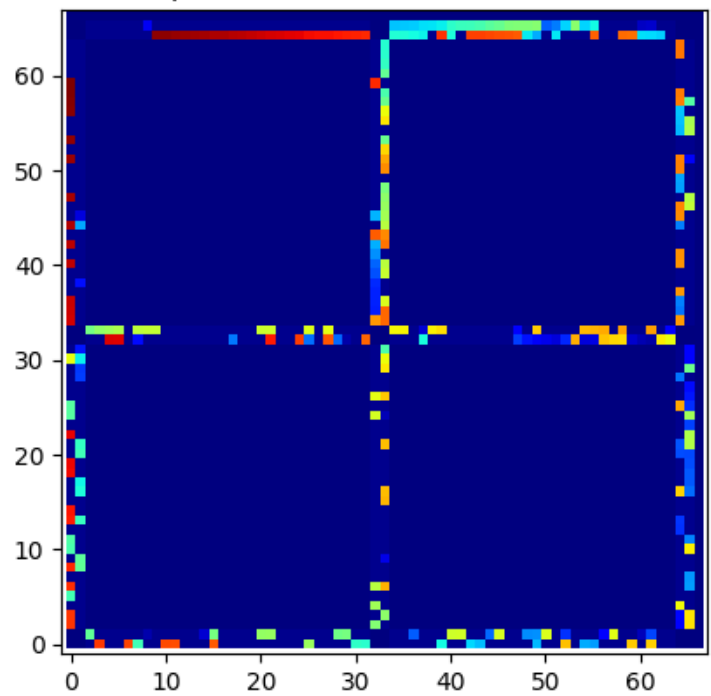
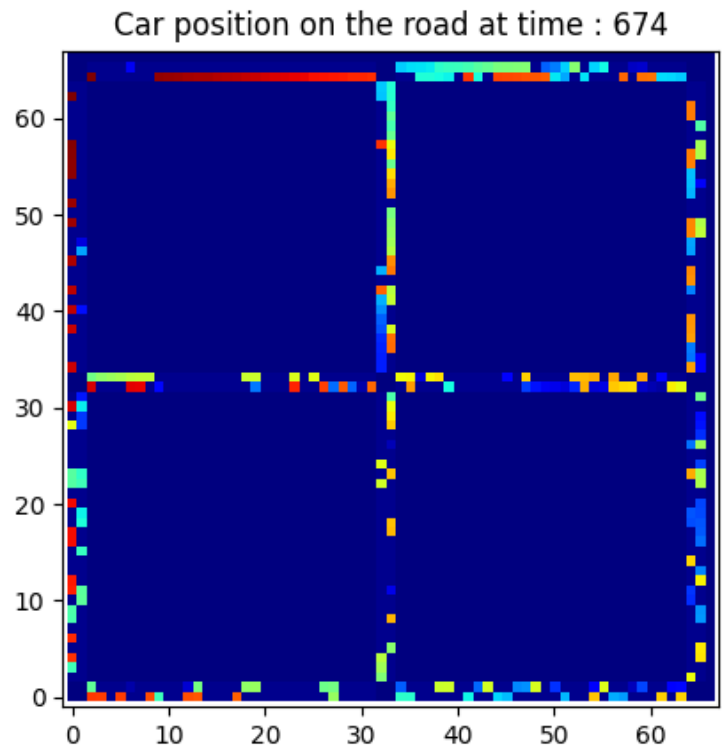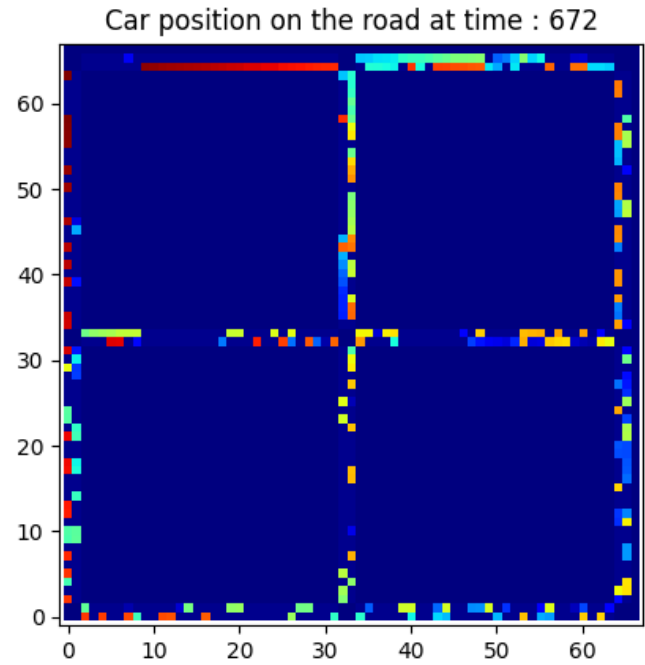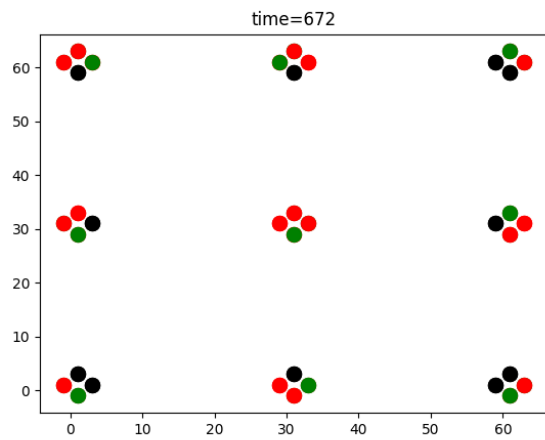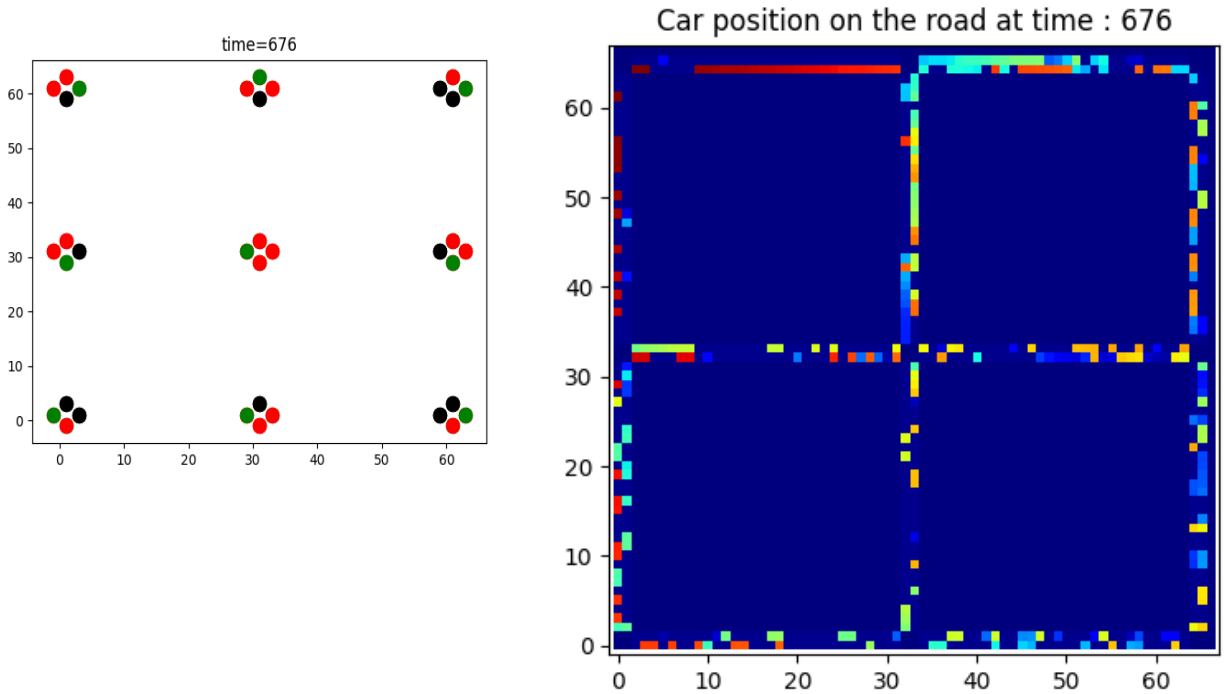4. After integrating the vehicle and infrastructure implementation, efforts were made to optimize the system to achieve maximum throughput.
We have considered different cases to analyze the system performance for different car generation rate, signal switching frequency.
We also tried to optimize the system by implementing effective car generation algorithm.

Case with car generated at every instance and signal switching:

Time after which signal will change (in s)	Total cars	Total completed cars	Throughput (%)
4	738	650	88
6	750	617	82
8	736	612	83
10	750	570	76
12	732	589	79
14	723	580	80
16	732	575	78
18	710	577	81
20	679	551	81
22	662	568	85

The throughput was maximum (88%) when car is generated at every time step i.e. 2 seconds and signals were switching every 4 secs.

The below are the output for top 3 maximum throughput cases:

```
74        priority_signal();
75
76
77        if(t%4 == 0){
78        controller(t);
79        }
80
81
82
83
84
85        //std::cout<<"Break 2";
```

```
17,734,18,6,11,0,20,18,5,3,20,18,6,8,23,21,1,
17,735,23,26,14,15,13,11,23,21,2,4,6,8,23,17,14,
17,736,10,17,14,15,13,11,23,21,2,4,6,8,10,12,14,
17,737,10,1,14,15,13,11,23,21,2,4,6,8,10,12,14,
17,738,-1,-1,-1,0,20,18,5,3,20,18,6,8,23,17,14,
Total completed cars = 650
Total collisions in a slot = 0
U turn violations = 0
R violations = 0
Misdirection = 0
Total crashes = 0
Throughput = 88
```

```
73        }
74        priority_signal();
75
76
77        if(t%6 == 0){
78        controller(t);
79        }
80
81
82
```

```
17,749,6,23,12,15,13,11,23,21,2,4,6,8,23,21,1,
17,750,-1,-1,-1,15,16,21,2,4,6,8,10,12,16,21,1,
Total completed cars = 617
Total collisions in a slot = 0
U turn violations = 0
R violations = 0
Misdirection = 0
Total crashes = 0
Throughput = 82
```

```
72        }
73
74        priority_signal();
75
76
77        if(t%8 == 0){
78        controller(t);
79        }
80
81
82
```

```
17,734,3,13,9,0,20,18,5,3,20,18,6,8,23,21,1,
17,735,6,23,12,15,13,11,23,21,2,4,6,8,23,17,14,
17,736,-1,-1,-1,15,13,11,23,21,2,4,6,8,10,12,14,
Total completed cars = 612
Total collisions in a slot = 0
U turn violations = 0
R violations = 0
Misdirection = 0
Total crashes = 0
Throughput = 83
```

```
74        priority_signal();
75
76
77        if(t%22 == 0){
78        controller(t);
79        }
80
81
82
```

```
17,662,-1,-1,-1,0,2,4,19,17,13,11,9,7,19,17,14,
Total completed cars = 568
Total collisions in a slot = 0
U turn violations = 0
R violations = 0
Misdirection = 0
Total crashes = 0
Throughput = 85
```

Case with Efficient car generation and signal switching

As efforts to optimize the system, we tried to implemented the efficient car generation algorithm. In order to generate maximum cars, a car was generated every 2 seconds for first 500seconds of the simulation. Because, the minimum time a car can take to return the destination is 484 seconds by travelling shortest path of 8 segments. Hence, we can keep the signal always Green for the junction 0 and signal 1 for the car generated just now since, for the first 480 seconds of simulation no car will return. We also have given priority signals for returning cars for last 600

seconds and stop generating new cars because the car generated after 600 seconds will not reach the destination as the simulation is running only for 1 hour.

We tried calling the signal_controller for a variety of switching times with efficient car generation. Below are the results:

Using this we were able to achieve 92% throughput with efficient car generation and signal switching at every 8 seconds.

Time after which signal will change (in s)	Total cars	Total completed cars	Throughput (%)
6	701	598	85
8	655	604	92
10	692	586	84
12	675	563	83
14	716	576	80
16	609	558	91
18	683	541	79
20	699	522	74
22	635	553	87

```
72      }
73
74          priority_signal();
75
76
77          if(t%8 == 0){
78          controller(t);
79          }
80
81
82
83
```

```
17,652,6,13,12,0,20,18,5,3,20,18,6,8,10,12,14,
17,653,23,18,14,15,13,11,23,21,2,4,6,8,23,17,14,
17,654,23,17,14,15,13,11,23,21,2,4,6,8,23,21,1,
17,655,-1,-1,-1,0,20,18,5,3,20,18,6,8,23,21,1,
Total completed cars = 604
Total collisions in a slot = 0
U turn violations = 0
R violations = 0
Misdirection = 0
Total crashes = 0
Throughput = 92
```

```
74          priority_signal();
75
76
77          if(t%16 == 0){
78          controller(t);
79          }
80
81
82
```

```
13,608,1,20,12,15,13,11,9,7,5,3,1,
17,609,-1,-1,-1,15,13,11,23,21,2,4,6,8,23,17,14,
Total completed cars = 558
Total collisions in a slot = 0
U turn violations = 0
R violations = 0
Misdirection = 0
Total crashes = 0
Throughput = 91
```

```
73
74          priority_signal();
75
76
77          if(t%22 == 0){
78          controller(t);
79          }
80
81
82
83
84
85          //std::cout<<"Break 2";
86          if(t<500){
87            il[0][0]=1;
88            il[0][1]=2;
89            il[0][2]=0;
```

```
13,626,1,21,12,15,13,11,9,7,5,3,1,
17,627,8,19,13,15,13,11,23,21,2,4,6,8,23,21,1,
19,628,22,26,9,0,2,4,19,22,10,12,16,18,6,8,10,12,14,
17,629,18,15,11,0,20,18,5,3,20,18,6,8,23,21,1,
17,630,18,14,11,0,20,18,5,3,20,18,6,8,23,17,14,
17,631,6,6,12,15,13,11,23,21,2,4,6,8,23,17,14,
13,632,1,5,12,15,13,11,9,7,5,3,1,
17,633,10,27,12,15,16,21,2,4,6,8,10,12,16,21,1,
17,634,6,5,12,15,13,11,23,21,2,4,6,8,23,21,1,
13,635,-1,-1,-1,0,2,4,6,8,10,12,14,
Total completed cars = 553
Total collisions in a slot = 0
U turn violations = 0
R violations = 0
Misdirection = 0
Total crashes = 0
Throughput = 87
```

Integration of the main() function

```cpp
#include <iostream>
#include <time.h>
#include <queue>
#include <vector>
#include <map>
#include <string>
#include "segment.h"
#include "crash.h"
#include "controller.h"
#include "signal.h"
using namespace std;

int main() {
    int w=0;
    int b=1;
    //Initializations of global variables
    t=0;
    total_cars = 0;
    total_completed_cars = 0;
    total_cars = 0;
    u_turn_vio = 0;
    r_vio = 0;
    mis_dir_cnt = 0;

    //Initialization functions to build the road system and bring it to initial
state
    create_segments();
    create_possible_routes();
    car_pos_signal_map();

    //Initialization function from the I-group
    initialise_queue ();
    initial_signal_priority();
    priority_signal();
    while(t<3600){
      init_crash_count = 0;
      if( t > 0){
        int car_gen_factor = 4;
        if(total_cars == 0 || car[total_cars-1][2] != -1){
          if(t % car_gen_factor == 0){
```

```cpp
                generate_car(total_cars);
                total_cars = total_cars + 1;
                }
            }
        }

        priority_signal();
        if(t%14 == 0){
        controller(t);
        }
        red_vio_pre_check();
        u_turn_precheck();
        junction_pre_check_for_crash();
        update_all_car_pos();
        junction_pos_check_for_crash();
        u_turn_poscheck();
        red_vio_pos_check();
        r_vio = r_vio + check_r_vio();
        u_turn_vio = u_turn_vio + check_utrun_vio();
        total_crash = total_crash + check_crash();

        t=t+1;
        t=t+1;
        w = w+1;
    }

    filled_slot_info(t);
    final_pos_car();
    rem_cars = car_remaining_in_grid();
    collision_in_a_segment(total_cars,total_completed_cars+rem_cars);
    no_col_in_slot = 0;
    std::cout<<"Throughput = "<<(total_completed_cars*100)/total_cars<<'\n';

    clear();

return 0;
}
```